IBM

System i

システム管理
実行管理

バージョン 6 リリース 1

IBM

System i

システム管理
実行管理

バージョン 6 リリース 1

ご注意

本書および本書で紹介する製品をご使用になる前に、 251 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i5/OS (製品番号 5761-SS1) のバージョン 6、リリース 1、モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。 このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

IBM 発行のマニュアルに関する情報のページ

http://www.ibm.com/jp/manuals/

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： System i
Systems management
Work management
Version 6 Release 1

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2008.2

目次

実行管理機能

実行管理機能は、i5/OS® オペレーティング・システムにおける重要なコンポーネントです。

すべての作業はこの機能によってシステムに入れられ、System i™ ナビゲーター製品で処理され、実行され、完了されるため、実行管理機能はシステムの基盤と言えます。実行管理機能は、単純なバッチ・ジョブを週ごとに実行するか、(Lotus Notes® のような) アプリケーションを毎日呼び出すかに関係なく、システムで実行されるジョブおよびオブジェクトの管理を可能にします。さらに、システム操作を制御したり、必要な場合にアプリケーションにリソースを割り振ったりするのに必要なコマンドおよび内部機能をサポートします。

System i 製品はすでにセットアップされ、すぐに使用できる状態になっています。ほとんどのユーザーはデフォルト設定を変更する必要がないでしょう。しかし、実行管理機能の一部を貴社の状況に合わせて調整する必要がある場合は、この機能に関連した用語や概念、さらにはシステムのパフォーマンスを最善のものとするための統合方法について理解しておく必要があります。

この一連のトピックは、System i の経験者か初心者かを問わず、実行管理機能を分かりやすく概観するのに役立ちます。このトピックにはさまざまな入り口点が含まれているので、実行管理機能についてどこから学び始めるかを選択することができます。

注: また、System i ナビゲーター・タスクを使用することによって、実行管理機能を Web 上で処理することができます。そのため、実行管理機能の諸機能を Web ブラウザーで処理することも可能になっています。詳しくは、System i ナビゲーター (Web 対応) を参照してください。

実行管理の PDF ファイル

この情報の PDF ファイルを表示および印刷することができます。

- 本書の PDF バージョンを表示またはダウンロードするには 実行管理を選択します。
- 「実行管理」の Advanced Job Scheduler の部分のみの PDF バージョンを表示またはダウンロードするには、Advanced Job Scheduler を選択します。

次の関連トピックを表示またはダウンロードできます。

- パフォーマンスには、以下のトピックが含まれています。
  - パフォーマンスの計画
  - システム・パフォーマンスの管理
  - パフォーマンス管理用のアプリケーション
- マネージメント・セントラルには、1 つ以上のシステムにわたるシステム管理タスクを同時に実行する上で役に立つ情報が含まれています。

PDF ファイルの保管

表示または印刷のために PDF をワークステーションに保管するには、以下のようにします。

1. ご使用のブラウザーで該当の PDF リンクを右クリックする。
2. PDF をローカルに保存するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。

4. 「**保存**」をクリックする。

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe® Reader がシステムにインストールされている必要があります。 このアプリケーションは、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) から無料でダウンロードできます。

実行管理機能の概要

実行管理機能は、システム操作およびシステム上の日常のワークロードを制御するのに必要なコマンドおよび内部機能をサポートします。さらに、実行管理機能には、アプリケーション用のリソースを配布してシステムがアプリケーションを処理できるようにするために必要な機能が含まれています。

システムの目的は作業を実行することです。作業はシステムに入り、処理され、そして終了します。実行管理機能をこの 3 つの語で捉えると、実行管理機能について理解しやすくなります。実行管理機能は、作業がどこからシステムに入るのか、作業がどのリソースを使ってどこで処理されるのか、および作業の出力の行き先について記述しています。

実行管理機能についてあまりよく知りませんか? これ以降のトピックでは、実行管理機能の概要をさまざまな面から取り上げます。システムに関する予備知識にかかわりなく、実行管理機能の基本的な原理をしっかりと理解できるはずです。

ビジネスとしてのシステム

実行管理機能の総合的な概念をより簡単に把握するために、システムをビジネスになぞらえてみましょう。

単純システムは中小規模ビジネスに、複合システムはショッピング・モールに、それぞれなぞらえることができます。手製の木製家具を作成するビジネスを営む、小規模なストアがあると想定します。*作業の開始* として、小さなテーブル、いす、本棚が注文されます。*作業の処理* として、製作者は顧客に電話をかけて注文を確認し、スタイル、サイズ、および色などを含む設計ポイントの相談を受けます。製作者は家具の各部を設計し、必要な材料を集め、その家具を作成します。家具の完成後に、それは納品され、*作業の完了* となります。

複合システムは、多くの単純システムの組み合わせであるので、複合システムになぞらえることができる例として、1 領域に多くの小規模/大規模ビジネスがあるショッピング・モールがあります。製作者はモールの北西の角でビジネスを営み、パン屋は東側通路に面した場所でビジネスを営みます。パン屋と製作者は入力および出力において異なります。つまりその注文と製品はかなり異なるものです。加えて、それぞれのビジネスでその作業を処理するためにかかる時間もかなり異なっており、それぞれの顧客もそのことを知り、理解しています。

実行管理機能の用語

複合システム (ショッピング・モール) は、多くの単純システム (ストア) の寄せ集めです。これらの単純システムは、サブシステム と呼ばれます。

ビジネス内の作業の構成部分はジョブ と見なされます。作業の構成部分の例には、顧客からの手紙、電話、注文、夜間の清掃などがあります。同じことが System i 製品についても言えます。システム上で、各ジョブには固有の名前があります。

ジョブ記述 は、サブシステムに送られてくる作業の処理方法を記述しています。ジョブ記述には、ユーザー ID、ジョブ待ち行列、および経路指定データなどのさまざまな情報が含まれています。ジョブ記述内の情報は、中小規模ビジネスのジョブの記述になぞらえることができます。

ビジネスの概観 どのストアにも、青写真、つまりストアの計画があります。これらの計画は実際には、ビジネスの物理的な構成要素のさまざまな詳細を記した単なる記述です。ビジネスによってはそのストアは、2 つのフロア、5 つのドア、3 つの郵便受け、2 台の電話を備えているかもしれません。システム上では、*サブシステム記述* には、サブシステムについてのすべての情報が含まれます。

作業の依頼元 製作者の場合、作業は顧客からの電話、問い合わせ、および店に訪れる人物などにより依頼されます。システム上では、作業は多くの場所から依頼される可能性があります。この例には、ジョブ待ち行列、ワークステーション、通信、自動開始ジョブ、および事前開始ジョブなどが含まれます。

スペースの確保 モール内では、それぞれのビジネス (サブシステム) は一定量のフロア・スペースを持ちます。システム上では、メモリー・プール により、それぞれのサブシステム (ビジネス) がその作業に着手する主記憶域 (またはフロア・スペース) を制御できます。 ストア (サブシステム) にさらに広いフロア・スペースがあれば、さらに多くの顧客、またはジョブをストア内に収容することができます。

作業の送信方法 必要なストアを見つけられない顧客は、正しい方向に進むために情報ブースを使用できます。同じことはご使用のシステムにも当てはまります。*経路指定項目* は、ストア・ディレクトリーまたは情報ブースと似ています。経路指定項目を見つけた後に、それはジョブを正しい場所に誘導します。ただし、経路指定項目をまず見つける必要があります。これは*経路指定データ* を使用して実行します。経路指定データは、正しい経路指定項目を見つけるためにジョブが使用するものです。

作業の処理方法 製作者は、それぞれのジョブに対して優先順位を付ける必要があります。週末が納期のいすは、月末が納期の本棚の前に完成させる必要があります。システム上では、クラス はサブシステム内にある間にジョブの処理方法についての情報を提供します。この情報には、実行時の優先順位、最大記憶域、最大 CPU 時間、およびタイム・スライスが含まれます。このそれぞれの属性は、ジョブの処理方法およびいつ処理されるかに関係してきます。

モール内のすべてのストアに影響する規則があるのと同様に、システム上のすべてのサブシステムに影響する規則があります。この規則の例には、システム*値* があります。システム値は、システム全体に適用される一連の情報です。システム値には、日時、構成情報、サインオン情報、システム・セキュリティー、および記憶域処理などの情報が含まれています。

モール内の各顧客は、それぞれに固有の情報を持ちます。システム上では、*ユーザー・プロファイル* に、特定ユーザーに固有の情報が保持されています。顧客のクレジット・カードと同様に、ユーザー・プロファイルは、そのユーザーに特定権限を付与し、そのユーザーのジョブのユーザー属性を割り当てます。これらの*ジョブ属性* は、ジョブ記述、出力待ち行列またはプリンター、メッセージ待ち行列、アカウンティング・コード、スケジューリング優先順位、およびその他が含まれる情報を提供します。

ジョブの開始から終了まで

System i の実行管理機能の基本を理解するために、一般的なバッチ・ジョブがシステム内でどのように処理されるかを見てください。

一般的なバッチ・ジョブの開始から終了までは、それがシステムに投入された時点で開始されます。その後、ジョブはジョブ待ち行列に入れられ、サブシステムに渡されて実行されるのを待ちます。ジョブがサブシステムに渡された後、実行のためのメモリーが割り振られます。その後、プリンター出力ファイル (スプール・ファイルとも呼ばれる) は出力待ち行列に送られ、その後の実行指示 (たとえば、印刷など) を待ち

ます。すべてのジョブがこの流れに従うわけではありませんが、この一般的なジョブのライフ・サイクルにより、その他の作業がシステム上でどのように完了するかをより良く理解できます。

ジョブを投入する → ジョブがジョブ待ち行列に入れられる → ジョブがサブシステムに入れられる → メモリー・プールが、サブシステムにメモリーを割り振る → ジョブが完了し、出力待ち行列に移動する

ジョブの投入

ジョブを投入すると、ジョブが作成され、システムに入れられます。この時点で、属性がジョブに与えられます。

ジョブ記述には、ジョブが実行管理のライフ・サイクルを通じて使用する*属性*が含まれています。これらの属性には、ジョブが実行を開始するユーザー・プロファイル、要求データ (何を行うかをジョブに知らせる)、ライブラリー・リストの初期ユーザー部分などが含まれます。さらにジョブ記述には、ジョブが入れられるジョブ待ち行列および経路指定データも入っています。経路指定データは、サブシステムが後でジョブの実行を開始するために必要な情報の入った経路指定項目を検索するために使用されます。さらに、ジョブ記述には出力待ち行列も定義されます。ジョブからのプリンター出力 (スプール・ファイルとも呼ばれる) の出力先が記述されています。

ジョブがジョブ属性の値 (初期設定、カスタマイズ) を受け取ると、ジョブはジョブ待ち行列に移動してサブシステムに入れられるのを待ちます。

ジョブがジョブ待ち行列に入れられる

*ジョブ待ち行列*は、サブシステムに入れられるバッチ・ジョブにとっての入り口です。サブシステムの「待合室」と考えることができます。

いつジョブがジョブ待ち行列から取り出されてサブシステムに入れられるかには、ジョブ待ち行列でのジョブ優先順位、ジョブ待ち行列の順番、最大アクティブ・ジョブなど、さまざまな要素が影響します。これらの要素が相互作用した結果、ジョブはジョブ待ち行列から取り出されてサブシステムでの実行が開始されます。

ジョブがジョブ待ち行列に入れられると、それはジョブ待ち行列が割り当てられているサブシステムから使用可能になります。サブシステムは複数のジョブ待ち行列を入れることができる (しかし、ジョブ待ち行列は複数のサブシステムに送ることはできない) ため、サブシステムでの順番はサブシステムがいつジョブ待ち行列を処理するかを判別できます。サブシステムは、ジョブ待ち行列内のジョブのジョブ優先順位の前に、ジョブ待ち行列の順序を確認します。サブシステムは、ジョブ待ち行列の優先順位を使用して、ジョブ待ち行列の他のジョブとの関係を考えてジョブをいつ投入するかについて決定します。ジョブ優先順位と最大アクティブ・ジョブによって、いつジョブがサブシステムに入れられるかが判別されます。

ジョブがサブシステムに入れられる

サブシステムとは、システムがジョブの使用するリソースを管理し、実行されるジョブを制御する操作環境のことです。ジョブがサブシステムで実行されると、そのサブシステム・ジョブが、ジョブに関するユーザーの要求 (たとえば、ジョブの保留、解放、終了など) を実行します。ジョブはサブシステムに入れられると、アクティブ状態になります。

ジョブと同様、サブシステムにも記述が用意されており、それによって作業を完了するのに必要な重要情報がやり取りされます。サブシステム記述には、経路指定項目があります。この*経路指定項目*は、実行時環境を制御する属性の入ったクラス・オブジェクトを参照します。しかし、ジョブが経路指定項目を取得する前に、経路指定データを経路指定項目の比較値と突き合わせる必要があります。この関連づけが行われないと、ジョブは実行されません。

経路指定データと経路指定項目の間で関連づけが行われると、ジョブが使用するクラス・オブジェクトが決まります。実行時環境を制御する属性には、実行優先順位、タイム・スライス、最大待ち時間、最大処理時間、最大一時記憶域、およびスレッドの最小数が含まれます。

サブシステム記述は、サブシステムに割り振られるメモリー・プールを定義します。 さらに、サブシステム記述には、最大アクティブ・ジョブが含まれていますが、これはサブシステム内で一度にアクティブ状態にできるジョブの最大数です。

ジョブはアクティビティー・レベルを取得して、メモリー・プールを割り振られるまで、実行できません。ジョブ記述のようにサブシステム記述によって、使用するメモリー・プール、経路指定項目、最大アクティブ・ジョブ、サブシステム内のアクティブ・ジョブの現行の数などの情報を送信します。

サブシステムはメモリー・プールからのメモリーを使用して、ジョブを実行する

メモリーは、サブシステムがジョブの実行のために使用する、メモリー・プールからのリソースです。メモリー・プール内のメモリーの量とともに、その他のジョブがどれほどメモリーのために競合するかということも、ジョブの実行効率に影響します。

メモリー・プールは、ジョブに実行のためのメモリーを提供します。ジョブがメモリー・プールで実行される方法には、メモリー・プールのサイズやメモリー・プール内のアクティビティー・レベル、さらにはページングや障害など、様々な要素が影響します。メモリー・プールでのアクティビティー・レベルは、メモリー・プール内で一度に実行できるスレッドの数に直接影響を及ぼします。すべてのジョブに少なくとも 1 つのアクティブ・スレッドがありますが、複数のスレッドを持つことができるジョブがあることも覚えておいてください。複数のスレッドがジョブに与えられることによって、一度に複数のことを行うことができます。たとえば、あるスレッドがたくさんのデータの処理を待っている間、別のスレッドが呼び出されて計算を行うことも可能です。

ページングは、同期および非同期で実行されるメモリー内外へのデータの移動のことです。ページが変更されない場合、ページは記憶域に書き出されるか、書き出さずにメモリーから除去されます。障害が発生すると、サーバー上でページングが起こります。参照ページ、またはデータの一部がメモリーにない場合に障害が発生します。これにより、プログラムはそのデータがページに入れられるのを待つ必要があるため、その実行は停止します。

サブシステムは、その中のタイプの異なるジョブを実行する場合には異なるメモリー・プールを使用します。

ジョブが終了して出力待ち行列に入れられる

ジョブのプリンター出力 (スプールされたファイルとも呼ばれる) は、出力待ち行列に送られ、そこでプリンターまたはファイルに送られるのを待ちます。出力待ち行列はジョブ待ち行列と似ていて、出力をどのようにプリンターに送信するかを制御します。出力待ち行列によって、どのファイルが最初に印刷されるかを制御することができます。

*出力待ち行列*とは、プリンター出力ファイルが処理され、プリンターに送信されるのを待機する領域です。プリンター出力は、システム、または印刷ファイルを使用するユーザーのいずれかによって作成されます。印刷ファイルは、プリンター出力の属性のデフォルト値が設定されているテンプレートまたはガイドラインのようなものです。これは、プリンター出力のライフ・サイクルの始まりです。

印刷ファイルには、プリンター出力を送信する方法を指定する、出力待ち行列 (OUTQ) 属性およびプリンター (DEV) 属性が含まれます。デフォルト設定は通常 *JOB です。この設定では、プリンター出力を送信する方法は、出力待ち行列およびプリンターのジョブ属性によって判別されます。設定されている出力待ち行列およびプリンターのジョブ属性は、ジョブの作成時に入手される情報に基づいています。これは、ジョ

ブを実行しているユーザーのユーザー・プロファイル、ジョブ記述、ワークステーション装置記述、およびデフォルト・プリンター (QPRTDEV) システム値からの情報に基づいています。

プリンター出力が作成可能な状態になると、システムは印刷ファイルおよびジョブ属性を (この順序で) 検査して、プリンター出力を処理する出力待ち行列、およびシステムが使用するプリンターを確認します。特定の出力待ち行列が検出できない場合、プリンター出力は QGPL/QPRINT に送られます。

プリンター出力ファイルが印刷可能な状態になった後、書き出しプログラム・ジョブ (出力待ち行列からプリンターに対してプリンター出力を処理するジョブ) は、プリンター出力ファイルからデータを取り出し、指定されたプリンターに送信します。

作業はどのように処理されるか

このトピックでは、作業とは何か、作業を開始する前に何をセットアップする必要があるか、作業はシステム内でどのように処理されるか、さらに作業の実行が終了するとどうなるかについて説明します。

作業とは何か

System i 製品上では、作業は、ユーザーが開始したものであれ、システムが開始したものであれ、常に実行されています。システム上で実行されるどのアクションにも、実行して完了される何らかのタイプの作業があります。

作業は、システムの電源を入れたり、ファイルをオープンしたり、データベースを照会したりすると、実行されます。システム上の各作業はジョブによって実行されます。ジョブは、ユーザーの呼び出しを待つアプリケーションのように単純な場合もあれば、 1 時間ごとにシステム上のユーザー数をモニターする常時実行のシステム照会のように複雑な場合もあります。一部のジョブ (とりわけバッチ・ジョブと対話式ジョブ) にはジョブ記述が関係しており、このジョブ記述がジョブを実行する時間と場所を設定します。

ジョブは特定の機能を実行するプログラムから成っています。ジョブが実行する機能の数に限界はありません。ジョブには、作業の実行を完了しなければならないステップバイステップの命令が含まれています。ジョブを構成するプログラムは決まった順序で実行されます。(たとえば、プログラム A はプログラム B が始まる前に実行する必要があります。)スレッドは、ジョブが作業を完了するのを支援します。アクティブ・ジョブには少なくとも 1 つのスレッドが含まれています。1 つのジョブに複数のスレッドが含まれている場合、そのジョブは一度に 2 つ以上のことを行うことができます。たとえば、あるスレッドがたくさんのデータの処理を待っている間、別のスレッドが呼び出されて計算を行うことも可能です。

作業がシステムに入れられる前に何が起きるか

システム・ジョブを除くすべてのジョブは、サブシステムの中で実行されます。作業をアクティブ・サブシステムで開始するには、メモリー・プールと、少なくても 1 つの作業入り口点のソースを確立する必要があります。ジョブ待ち行列は、作業のソースの一例です。

System i 製品には、ジョブ待ち行列、サブシステム、そしてメモリー・プールのデフォルト・セットが用意されており、システムを起動したらすぐに作業を開始できるようになっています。

サブシステムとメモリー・プールの構成を調整して、System i 製品の機能およびパフォーマンスを最適化することができます。たとえば、ビジネスの成功にバッチ・ジョブが欠かせないようなケースでは、バッチ・ジョブにさらに多くのメモリーを割り振りたいと考えるかもしれません。あるいは、Qbatch サブシステムで一度に実行されるジョブの数を少なくして、バッチ・ジョブが最大限にリソースを使って実行できるようにする必要があると判断するかもしれません。さらに、特殊な作業を実行するために特別に設計されたジョブ待ち行列、サブシステム、およびメモリー・プールを作成できます。たとえば、Nightreps と呼ばれ

るジョブ待ち行列を作成して、夜間にバッチ・レポートを Nightrep と呼ばれるサブシステムに送るようにし、これらのバッチ・ジョブの実行専用のメモリーを割り振るようにすることができます。

作業はどのようにシステムに入れられるか

実行処理項目は、ジョブをどのソースからサブシステムに入れて、実行可能にできるかを示します。ジョブのタイプごとに、異なるタイプの実行処理項目が使用されます。

たとえば、ほとんどのバッチ・ジョブは、ジョブ待ち行列を使用してサブシステムに入れられます。ジョブ待ち行列項目のメカニズムによって、ジョブ待ち行列はサブシステムへの作業のソースとして定義されます。

実行処理項目は、サブシステム記述に保持されます。サブシステム記述に現在実行中の作業タイプの実行処理項目がない場合、そのジョブはサブシステムで実行できません。 IBM 提供のサブシステムのサブシステム記述にはデフォルトの実行処理項目が用意されています。これらサブシステムとともに提供されているデフォルトの実行処理項目には、すでに特定のジョブを実行するために割り当てられているものがあるということに注意してください。

作業はどのように処理されるか

システムが起動すると、サブシステム・モニター・ジョブが開始されます。 サブシステム・モニター・ジョブは、サブシステム内のジョブを制御します。さらに、作業の開始と終了、サブシステム内の作業に関するリソースの管理も実行します。

作業 (またはジョブ) は実行処理項目によってサブシステムに入れられ、アクティブ状態になって実行可能な状態になります。作業は、実行のためのメモリーがサブシステムに割り振られていないと完了しません。メモリーはメモリー・プールによってサブシステムに割り振られます。

サブシステム記述はどのように作業の処理に役立つか

ジョブと同じように、サブシステムにもサブシステム記述と呼ばれる記述があります。サブシステム記述には、サブシステム内で一度にアクティブ状態にできる作業に関する方法、場所、数などの重要な情報や、作業を実行するのに使用できるリソースに関する情報が含まれています。

経路指定項目
: 経路指定項目は、サブシステム記述内に存在し、そのジョブにどのプログラムを実行するか、ジョブをどのメモリー・プールで実行するか、さらにジョブの実行にどのクラス・オブジェクトを使用するかをサブシステムに通知します。

クラス・オブジェクト
: クラス・オブジェクトは、実行優先順位、デフォルトの待ち時間、タイム・スライス、その他の属性を定義します。実行優先順位は、ジョブがいつ実行のためにプロセッサー時間を取得するかを決定するため、重要です。 実行優先順位は 0 から 99 までで、0 が最高優先順位です。 (システムを実行するのがシステム・ジョブであるため、優先順位 0 はシステム・ジョブだけに与えられます。)

ジョブがサブシステムに入れられると、サブシステムは経路指定項目にある比較値と経路指定データの突き合わせに追われます。経路指定データと経路指定項目内の比較値が一致すると、その経路指定項目がジョブに割り当てられます。どの経路指定項目内でも一致しない場合、ジョブは終了します。

ジョブがサブシステム内で実行される場合に影響を及ぼす別の要素は、サブシステム内で一度にアクティブ状態にすることができるジョブの数 (サブシステム内の最大アクティブ・ジョブとも呼ばれます) です。サブシステム内での最大アクティブ・ジョブ数に達すると、既存のアクティブ状態にあるジョブの実行が完了

するまで、追加のジョブはサブシステムに入れられません。ジョブの実行のためには、メモリーがサブシステムに割り振られなければなりません。メモリー・プール・アクティビティー・レベルは、メモリー・プール内でアクティブにできるスレッドの数をシステムに通知します。 アクティブ・ジョブには少なくとも 1 つのスレッドが含まれていることを覚えておいてください。メモリー・プール・アクティビティー・レベルに達すると、ジョブは別のスレッドがアクティビティー・レベルの使用をやめるのを待つ必要があります。したがって、ジョブはサブシステム内でアクティブ状態になりますが、実行はされません。

注: サブシステムの 最大アクティブ・ジョブをメモリー・プールのアクティビティー・レベルと混同しないようにしてください。

システムにおいて作業はどのように終了するか

出力待ち行列は、ジョブ待ち行列と同様の働きをして、印刷される出力をスケジュールします。プリンター出力および出力待ち行列の両方が、情報の印刷に使用される属性を保持しています。

プリンター出力には、印刷を待機中の情報など、処理を待機中の出力データが保持されています。プリンター出力には、いつ印刷するかをスケジュールする際に使用される重要な情報も保持されています。 プリンター出力属性には、プリンター出力が存在する出力待ち行列、優先順位、状況、およびプリンター出力のスケジュールが含まれます。

出力待ち行列には、プリンター出力ファイルを処理する順序を判別するための独自の属性が含まれています。さらに、プリンター出力および出力待ち行列に変更を加えるために必要な権限も含まれています。

プリンター出力をプリンターに送信する準備が完了すると、それは書き出しプログラム・ジョブによって処理されます。書き出しプログラム・ジョブはプリンター出力からデータを取り出して、それを印刷できるように準備します。

概念

実行管理機能の使用経験がないとしても、実行管理機能ツールを何年も使用してきた経験をお持ちでも、これらの実行管理機能の概念は有用かもしれません。

システムの構造

System i 製品を受け取った後、システムに用意されているサブシステム、開始プログラムに変更を加える必要があるかどうか、また作業の際に使用するユーザー・インターフェースの種類について知りたいと思われるかもしれません。

システム出荷時のサブシステム

変更を行わずにそのまま使用できる IBM 提供のサブシステム構成は 2 つあります。

システム起動時にシステムが使用する構成は、制御サブシステム/ライブラリー (QCTLSBSD) システム値によって制御されます。デフォルト構成は以下のサブシステム記述から成っています。

サブシステム	説明
Qbase (制御サブシステム)	Qbase は対話式ジョブ、バッチ・ジョブ、および通信ジョブをサポートします。このサブシステムには自動開始ジョブがあり、Qusrwrk、Qserver、および Qspl サブシステムを自動的に開始します。
Qserver	これはファイル・サーバー・サブシステムです。
Qspl	これは読み取りプログラムと書き出しプログラムのジョブをサポートするスプール・サブシステムです。

Qsyswrk	これはシステム作業サブシステムです。このサブシステムには、システム起動時およびシステムが制限状態から回復したときに自動的に開始するシステム機能をサポートするジョブが含まれています。
Qusrwrk	これはユーザー作業サブシステムです。このサブシステムには、ユーザーの代行処理を行うためにサーバーによって開始されるジョブが含まれています。

IBM 提供のもう 1 つの構成は、次のサブシステム記述から成っています。

サブシステム	説明
Qctl (制御サブシステム)	Qctl には、 Qinter、 Qbatch、 Qcmn、 Qusrwrk、 Qserver および Qspl サブシステムを自動的に開始する自動開始ジョブがあります。
Qinter	これは、コンソールにある対話式ジョブ以外の、対話式ジョブをサポートするサブシステムです。
Qbatch	これはバッチ・ジョブをサポートするサブシステムです。
Qcmn	これは TCP/IP 通信ジョブを除いた通信ジョブをサポートするサブシステムです。i5/OS システムでサポートされるさまざまな通信プロトコルには、これらの通信ジョブが必要です。
Qserver	これはファイル・サーバー・サブシステムです。
Qspl	これは読み取りプログラムと書き出しプログラムのジョブをサポートするスプール・サブシステムです。
Qsyswrk	これはシステム作業サブシステムです。このサブシステムには、システム起動時およびシステムが制限状態から回復したときに自動的に開始するシステム機能をサポートするジョブが含まれています。
Qusrwrk	これはユーザー作業サブシステムです。このサブシステムには、ユーザーの代行処理を行うためにサーバーによって開始されるジョブが含まれています。

Qbase 構成では、Qctl 構成で実行可能なすべての機能を実行することができ、しかもより少ないサブシステムで構成されているので管理が容易です。

Qctl デフォルト構成では、システム・アクティビティーをそれぞれのタイプに応じて種々のサブシステムに類別することによって、システム操作をより個別化して制御を実行できます。たとえば、週末にバッチ・ジョブを実行する必要があり、しかもその間はコンソール以外からはだれもサインオンできないようにしたい場合、Qctl 構成では Qinter サブシステムを終了することによってこれを容易に行うことができます。

ユーザー独自のサブシステム構成を作成することを検討しているのであれば、Qbase 構成ではなく Qctl 構成を出発点として使用したほうが作業が簡単になります。

開始プログラム

QSTRUPPGM は開始プログラムです。これは、制御サブシステムの開始時に自動開始ジョブから呼び出されるプログラムの名前を指定するシステム値です。このプログラムは、サブシステムおよびプリンターの開始などのセットアップ機能を実行します。このシステム値を変更できるのは、機密保護担当者または機密保護担当者権限のあるユーザーだけです。このシステム値に対する変更は、IPL を次回に実行するときに有効になります。

QSTRUPPGM には、以下の 3 つの値があります。

- QSTRUP QSYS: 指定したプログラムは、制御サブシステムの自動開始ジョブから制御が渡されることによって開始します。

- *NONE: 自動開始ジョブは、プログラムを呼び出すことなく正常に終了します。

関連情報

IPL を制御するシステム値

IPL の時に何が行われるか: デフォルトの開始プログラム QSYS/QSTRUP は、以下を行います。

- スプールされた作業用に QSPL サブシステムを開始します。
- QS36MRT および QS36EVOKE ジョブ待ち行列が保留されていた場合は、これらの待ち行列を解放します (これらは、System/36™ 環境で使用されます)。
- 操作援助機能クリーンアップを開始します (許可されている場合)。
- すべての印刷プログラムを開始します (ユーザーが IPL オプション画面で印刷プログラムを開始しないよう指定してある場合を除く)。
- QSERVER および QUSRWRK サブシステムを開始します。
- 制御サブシステムが QCTL の場合、QINTER、QBATCH、および QCMN サブシステムを開始します。

始動のタイプ

初期プログラム・ロード (IPL) 時に、システム・プログラムが、システム補助記憶域内の指定のロード・ソース装置からロードされます。 システム・ハードウェアも検査されます。i5/OS の制御パネルは、一連のシステム参照コードを表示します。これは、現在の状況を示し、問題があれば警告を出します。 IPL が終了すると、文字ベース・インターフェースによってサインオン画面が表示され、ユーザーは System i ナビゲーター にサインオンできるようになります。

システムは、いくつかの方法で開始できます。以下のようにできます。

- 構成変更を行わずにシステムを開始する。これは*不在 IPL* と呼ばれます。
- IPL 時にシステム構成を変更する。これは*在席 IPL* と呼ばれます。

  在席 IPL では、IPL オプション画面で選択するオプションに従って、さまざまな追加画面が表示されます。例えば、システム値および他のシステム属性を IPL 時に変更する画面、アクセス・パスを再構成する画面、物理ファイル制限の状況を検査する画面、新しい装置を構成および命名する画面、操作環境のオプションを指定する画面などです。
- システム制御パネルから IPL のタイプを変更する。
- システムのシャットダウンおよび再始動をスケジュールする。

IPL 時の一般的な問題は、*異常 IPL* と呼ばれます。

IPL およびシステム・シャットダウンについての詳細は、システムの開始と停止に関する情報を参照してください。

関連情報

システムの開始と停止

システムの電源遮断

システムの電源を切るときは注意する必要があります。あるタスクが完了せずにシステムの電源を切ると、データが損傷するか、またはシステムが予測不能な仕方で動作する可能性があります。

Information Center の以下のトピックには、システムの安全な電源遮断についての詳細が含まれています。

- 統合 Windows® サーバーが存在する場合にシステムを安全にシャットダウンする方法
- 論理区画のあるシステムの電源遮断

- 電源遮断システム出口プログラム
- 電源オフ調整用の出口プログラム

関連情報

統合 Windows サーバーが存在する場合に System i ハードウェアをシャットダウンする

論理区画のあるシステムの電源遮断

電源遮断システム出口プログラム API

電源オフ調整用の出口プログラム API

System i ナビゲーター

System i ナビゲーター は、Windows クライアント用の優れたグラフィカル・インターフェースです。System i ナビゲーター を使用すると、システムを Windows デスクトップから管理して実行できます。System i ナビゲーターを使用すると、実行管理機能に関連したほとんどのタスクを成し遂げることができます。

このインターフェースは、ユーザーの効率を高めるために設計されています。ですから、皆さんをガイドするオンライン・ヘルプを備えたSystem i ナビゲーター をご使用になるようお勧めします。 このインターフェースは発展途中にあり、ご自分のタスクの一部を実行するには PC5250 などの従来のエミュレーターを使用する必要が依然ある場合もあります。そのようなタスクについてトピックで扱われている場合には、そのトピックの説明ステップ内で文字ベース・インターフェースを使用するように勧められます。

関連情報

System i ナビゲーターの理解

System i ナビゲーター (ワイヤレス対応)

System i ナビゲーター・タスク (Web 対応)

サブシステム

サブシステムとは、作業が処理されるシステム上の場所のことです。サブシステムとは、システムがワークフローとリソース使用を調整するために使用する単一の事前定義された操作環境のことをいいます。システムには、それぞれ独立して作動する複数のサブシステムを含めることができます。サブシステムはリソースを管理します。

システム・ジョブを除くすべてのジョブは、サブシステムの中で実行されます。各サブシステムは、固有の操作を実行できます。たとえば、あるサブシステムは対話式ジョブだけを処理し、他のサブシステムはバッチ・ジョブだけを処理するように設定できます。また、複数のサブシステムで種々の作業を処理できるように設計することもできます。サブシステムの数および各サブシステムで処理する作業のタイプは、ユーザーが設定できます。

サブシステムの実行時特性は、サブシステム記述と呼ばれるオブジェクトに定義されています。たとえば、ジョブ待ち行列からサブシステムに送信される作業量 (ジョブの数) を永続的に変更する場合には、サブシステム記述でジョブ待ち行列項目を変更するだけで済みます。

関連タスク

179 ページの『共通サブシステム・タスク』
ここでは、サブシステム上で実行できる最も一般的なタスクについて説明します。

182 ページの『サブシステム記述の作成』
サブシステム記述は、2 とおりの方法で作成することが可能です。既存のサブシステム記述をコピーしたものを変更することができますし、全く新しい記述を作成することもできます。

関連情報

経験報告: サブシステム構成

制御サブシステム

制御サブシステムは、システムの開始時に自動的に開始される対話式サブシステムであり、システム・オペレーターがシステム・コンソールを使用してシステムの制御に使用するサブシステムです。これは、サブシステム/ライブラリー制御 (QCTLSBSD) システム値で識別されます。

IBM 提供の 2 つの完全な制御サブシステム記述として、QBASE (デフォルトの制御サブシステム) および QCTL があります。 どの時点でもシステム上では 1 つの制御サブシステムだけをアクティブにできます。

システムが制限状態の場合、システム上のほとんどのアクティビティーは終了しており、1 つのワークステーションだけがアクティブです。システム保管 (SAVSYS) や記憶域の再利用 (RCLSTG) などのコマンドを実行する場合、システムはこの状態でなければなりません。 装置の問題を診断するための一部のプログラムでも、システムが制限状態にあることが必要です。この状態を終了するには、制御サブシステムを再始動する必要があります。

注: 1 つのバッチ・ジョブだけをアクティブにできる、バッチ制限状態もあります。

制御サブシステムを含め、すべてのサブシステムが終了すると、制限条件が作成されます。それぞれのサブシステムは個別に終了させることも、または ENDSBS SBS(*ALL) OPTION(*IMMED) を使用することもできます。

重要: 制御サブシステム内に 1 つのジョブだけが残された状態になるまでは、システムは制限状態になりません。時には単一のジョブだけが残っているように思えても、システムが制限状態にならない場合があります。この場合は、中断されたシステム要求ジョブまたはグループ・ジョブ、あるいは切断されたジョブが、残りのアクティブな画面上にないことを検査する必要があります。アクティブ・ジョブ処理 (WRKACTJOB) コマンドを使用して、F14=「組み込み」を押し、中断または切断されたジョブを表示します。それらのジョブが存在する場合、システムが制限状態になるようにそれらを終了させる必要があります。この条件の検出時に、ENDSYS および ENDSBS 機能は、CPI091C 情報メッセージをコマンド発行者に送信します。

関連タスク

198 ページの『制御サブシステムの作成』
IBM 提供の 2 つの完全な制御サブシステム構成として、QBASE (デフォルトの制御サブシステム) および QCTL があります。 システム上では、一時点で 1 つの制御サブシステムだけをアクティブにできます。一般に、IBM 提供のサブシステム構成は、たいていのビジネス・ニーズには十分であるはずです。 ただし、制御サブシステムのユーザー固有のバージョンを作成したり、それを企業の固有のニーズにさらに厳密に合わせて構成することができます。

199 ページの『システムを制限状態にする』
制御サブシステムを含むすべてのサブシステムが終了する場合、システムは制限状態になります。対話式ワークステーションから以下の 2 つコマンドのいずれかを使用すると、システムを制限状態にすることができます。

関連情報

経験報告: 制限状態

複数のサブシステムについて考慮する理由

システム上のユーザー数が増えるにつれて、多くの場合、作業セットに対して 1 つのサブシステムでは不十分になってきます。ユーザーを複数のサブシステムに分割することには、いくつかの利点があります。

作業の管理容易性の改善

各サブシステムで実行されている作業をよりよく制御できます。例えば、サーバー・ジョブの場合、すべてのデータベース・サーバー・ジョブをあるサブシステムに、リモート・コマンド・サーバー・ジョブを別のサブシステムに、DDM サーバー・ジョブをさらに別のサブシステムにという要領で分離したい場合があるかもしれません。さらに、複数のサブシステムを使用することにより、独自のメモリー・プールを持つジョブのグループを分離できます。これにより、あるグループが他のジョブに悪影響を与えることはなくなります。

ダウン時間のユーザーに与える影響の減少

例えば、毎週金曜日の午後にバックアップを目的としてシステムを制限状態にする場合は、一度に 1 つのサブシステムを終了することにより、ユーザーを少しずつオフラインにできます。

スケーラビリティーおよび可用性の改善

単一のサブシステムが処理するユーザー数を減らすことにより、サブシステムが使用中になる頻度は少なくなり、処理する作業要求によりよく反応できるようになります。

対話式サブシステムでのエラー許容度の改善

作業を複数のサブシステムに分散することによって、ネットワーク障害が発生しても、複数のサブシステムが装置回復処理を管理できます。

対話式サブシステム起動時間の短縮

作業を複数のサブシステムに分割することにより、サブシステム起動時間を短縮できます。

パフォーマンス・チューニング用の追加オプション

複数のサブシステムを使用することにより、経路指定項目の数が少ないサブシステムをセットアップできます。

関連情報

経験報告: サブシステム構成

サブシステム記述

サブシステム記述は、システムによって制御される操作環境の特性を定義する情報を含んでいるシステム・オブジェクトです。このオブジェクト・タイプのシステム認識 ID は *SBSD です。サブシステム記述は、どれだけの作業が、どこから、どのようにサブシステムに入ってくるか、およびそれらの作業を実行するためにサブシステムがどのリソースを使用するかを定義しています。アクティブ・サブシステムの名前には、サブシステム記述の単純名が使用されます。

一連の青写真のように、各サブシステム記述は固有で、サブシステムを記述する具体的な特性を含んでいます。この記述には、作業がどこからサブシステムに入るか、サブシステムがどれだけの作業を処理できるか、どれほどの主記憶 (メモリー) が使用されるか、およびサブシステム内のジョブがどれほど速く実行されるかなどがあります。

システムで提供されるサブシステム記述を (変更して、または変更しないで) 使用することも、独自のサブシステム記述を作成することもできます。

関連タスク

187 ページの『サブシステム記述の変更』
サブシステム記述変更 (CHGSBSD) コマンドは、指定されたサブシステム記述の運用属性を変更します。サブシステムがアクティブである間にサブシステム記述を変更することができます。サブシステム記述を変更するには、文字ベース・インターフェースを使用します。

182 ページの『サブシステム記述の作成』
サブシステム記述は、2 とおりの方法で作成することが可能です。既存のサブシステム記述をコピーしたものを変更することができますし、全く新しい記述を作成することもできます。

サブシステム記述属性:

サブシステム記述属性は、システム全体の共通属性です。サブシステムを作成する場合の最初のステップは、サブシステム属性の定義です。

サブシステム属性には以下が含まれます。

- サブシステム記述の名前およびその保管先のライブラリー
- サブシステムが使用するすべてのメモリー・プール定義

  サブシステム定義には、最大で 10 のメモリー・プール定義を指定できます。サブシステム定義には以下が含まれます。
  - プール定義 ID: これは記憶域プール定義の、サブシステム記述内の ID です。
  - サイズ: これは KB 単位 (1 KB = 1024 バイト) の倍数で表される記憶域プールのサイズであり、プールが使用できる主記憶域の量です。
  - アクティビティー・レベル: これはプール内で同時に実行できるスレッドの最大数です。
- サブシステム内で同時にアクティブ状態にできるジョブの最大数
- サブシステム記述のテキスト記述
- サブシステムに割り振られているワークステーションのサインオン画面を表示するために使用する、サインオン画面ファイルの名前およびライブラリー。
- サブシステム・ライブラリー名。これはライブラリー・リストのシステム部分で他のライブラリーより前に入力する必要があるライブラリーを指定する場合に使用できます (このパラメーターによって 2 次言語ライブラリーを使用できます)。

さらにサブシステム記述には、サブシステムに対する権限レベルについての情報が含まれています。この情報はセキュリティー属性によって保持され、サブシステム記述のその他の属性で保管されることはありません。サブシステム記述の権限は、「オブジェクト権限表示 (DSPOBJAUT)」コマンドを使用して表示できます。

実行処理項目:

実行処理項目は、ジョブをどのソースからサブシステムに入れることができるかを示します。さまざまなタイプのジョブに、特定のタイプの実行処理項目が使用されます。実行処理項目は、サブシステム記述の一部です。

以下の情報は、さまざまなタイプの実行処理項目と、その管理方法を説明しています。実行処理項目には、自動開始ジョブ項目、通信項目、ジョブ待ち行列項目、事前開始ジョブ項目、およびワークステーション項目の、5 タイプがあります。

自動開始ジョブ項目:

自動開始ジョブ項目は、サブシステムの開始とともに開始される自動開始ジョブを示します。サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

サブシステムに関連付けられた自動開始ジョブは、サブシステムの各開始時に自動的に開始されます。制御サブシステム内の自動開始ジョブは、(IBM 提供の制御サブシステムのように) 他のサブシステムを開始するために使用できます。自動開始ジョブは、繰り返しの作業を実行するバッチ・ジョブです。

たとえば、IPL で直前のシステム終了が異常であったとセンスされた場合に特殊リカバリー・プログラムを呼び出すには、制御サブシステムのサブシステム記述に自動開始ジョブ項目を追加することができます。このプログラムは、直前のシステム終了状況 (QABNORMSW) システム値をチェックします。通常のシステム終了の場合、QABNORMSW の値は '0' です。異常なシステム終了の場合、QABNORMSW の値は '1' です。

関連タスク

183 ページの『自動開始ジョブ項目の追加』
自動開始ジョブ項目の追加には、文字ベース・インターフェースを使用します。自動開始ジョブは、関連づけられているサブシステムが開始するときに自動的に開始します。この種のジョブは、一般的に特定のサブシステムに関連している初期設定作業を実行します。さらに、自動開始ジョブは繰り返し作業を実行したり、同じサブシステム内の他のジョブに集中的なサービス機能を提供したりします。

188 ページの『自動開始ジョブ項目の変更』
事前に定義されている自動開始ジョブ項目に対して、別のジョブ記述を指定することができます。自動開始ジョブ項目を変更するには、文字ベース・インターフェースを使用します。

192 ページの『自動開始ジョブ項目の除去』
文字ベース・インターフェースを使用して、自動開始ジョブ項目をサブシステム記述から除去できます。

通信項目:

通信作業項目は、処理する通信ジョブのソースをサブシステムに示します。 ジョブ処理は、サブシステムが通信プログラム開始要求をリモート・システムから受け取り、その要求の適切な経路指定項目が検出されたときに開始されます。

パフォーマンス上の理由から、プログラム開始要求を受け取るたびに通信ジョブを開始する代わりに、リモート・システムからのプログラム開始要求を扱うように事前開始ジョブを構成することができます。通信バッチ・ジョブをシステム上で実行する場合、通信作業の作業項目が含まれるサブシステム記述がシステム上に存在している必要があります。

関連タスク

183 ページの『通信項目の追加』
それぞれの通信項目では、1 つ以上の通信装置、装置タイプ、またはプログラム開始要求を受け取ったときにジョブを開始するサブシステムのリモート・ロケーションを記述しています。 サブシステムは通信装置を、その装置が現在他のサブシステムまたはジョブに割り振られていない場合は割り振ることができます。現在割り振られている通信装置は、最終的には割り振り解除され、他のサブシステムで使用可能になります。通信項目をサブシステム記述に追加するには、文字ベース・インターフェースを使用します。

188 ページの『通信項目の変更』
既存のサブシステム記述内の既存の通信項目の属性は、文字ベース・インターフェースを使用して変更できます。

192 ページの『通信項目の除去』
文字ベース・インターフェースを使用して、通信項目をサブシステム記述から除去できます。このコマンドを実行する前に、除去中の通信項目を通じてアクティブ状態のすべてのジョブを終了しなければなりません。

ジョブ待ち行列項目:

サブシステム記述内のジョブ待ち行列項目は、サブシステムがジョブを受け取るジョブ待ち行列を指定します。サブシステムの開始時に、サブシステムはサブシステムのジョブ待ち行列項目内で定義されたそれぞれのジョブ待ち行列の割り振りを試行します。

たとえば、サブシステム記述 QSYS/QBASE 内のジョブ待ち行列項目は、ジョブがジョブ待ち行列 QGPL/QBATCH を使用して開始できることを指定します。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステム QBASE はその開始時に、待ち行列上のジョブを処理します。サブシステム記述では、同時に処理できるジョブ (バッチまたは対話式) の最大数を指定できます。ジョブ待ち行列からアクティブにできるジョブの数は、ジョブ待ち行列項目に指定されます。

関連タスク

184 ページの『ジョブ待ち行列項目の追加』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列から開始されるジョブはバッチ・ジョブです。ジョブ待ち行列項目は、文字ベース・インターフェースを使用して追加します。

189 ページの『ジョブ待ち行列項目の変更』
指定されたサブシステム記述内の既存のジョブ待ち行列項目は変更可能です。このコマンドは、サブシステムがアクティブでも非アクティブでも発行できます。サブシステム内のジョブ待ち行列項目を変更するには、文字ベース・インターフェースを使用します。

192 ページの『ジョブ待ち行列項目の除去』
文字ベース・インターフェースを使用して、ジョブ待ち行列項目をサブシステム記述から除去できます。ジョブ待ち行列項目がサブシステム記述から除去される時に、そのジョブ待ち行列上のジョブは待ち行列に残っています。現在アクティブ状態のいずれかのジョブがジョブ待ち行列から開始された場合には、ジョブ待ち行列項目を除去することはできません。

事前開始ジョブ項目:

事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

プログラム開始要求が事前開始ジョブに添付されている場合、事前開始ジョブのジョブ属性はサブシステムによって変更されません。しかし通常は、サーバー・ジョブはジョブ属性を、スワップされたユーザー・プロファイルのジョブ属性に変更します。

事前開始ジョブの変更 (CHGPJ) コマンドを使用すると、事前開始ジョブはジョブ属性の一部を (プログラム開始要求のユーザー・プロファイルに関連付けられたジョブ記述で指定された、または事前開始ジョブ項目で定められたジョブ記述で指定の) ジョブ記述のジョブ属性に変更できます。

サーバーの事前開始ジョブ:

事前開始ジョブ・モデルには、通常デーモン・ジョブまたは受話者ジョブと呼ばれる基本聴取ジョブが 1 つと、クライアント要求を処理する複数のサーバー・ジョブが存在します。デーモン・ジョブは、接続要求用のポートで listen します。新しい接続を受け取ると、デーモンは一般処理を幾らか行い、待機中の事前開始サーバー・ジョブにソケット記述子を渡します。

事前開始ジョブは、再使用できます。このジョブがあるクライアントの作業を完了すると、環境がリセットされ、ジョブは別のクライアントからの要求を処理できるようになります。

ユーザー・コードを実行するサーバー・ジョブ (たとえば、リモート・コマンド・サーバー) では、ジョブは通常再使用されません。それは、ユーザー・コードがジョブ内で何らかの変更を行い、新しいクライアン

ト用に環境を確実にリセットすることができない場合があるからです。サーバーでジョブを再使用する場合、ジョブの変更 (QWTCHGJB) API を使用して、クライアントの要求が完了してから既知の状態にジョブの属性を変更して戻すことができます。

事前開始ジョブ・モデルを使用するサーバーには、ホスト・サーバー (SMTP サーバー、PPP サーバー、DDM/DRDA サーバー、 SQL サーバーなど) が含まれます。

関連概念

246 ページの『事前開始ジョブの調査』
このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

ワークステーション項目:

対話式ジョブとは、ユーザーがディスプレイ装置にサインオンした時点で開始され、ユーザーがサインオフした時点で終了するジョブのことです。このジョブを実行するために、サブシステムはワークステーション項目またはユーザー・プロファイルに指定されているジョブ記述を検索します。

ワークステーション項目により、サブシステムは予期されるワークステーションにアクセスします。ワークステーションが使用可能な場合、サブシステムはサインオン画面をディスプレイに送信します。

注: 制御サブシステムのサブシステム記述には、コンソールのワークステーション項目が含まれていなければならず、その項目はタイプが *SIGNON でなければなりません。 (*SIGNON は AT パラメーターの値であり、ワークステーション項目追加 (ADDWSE) コマンド上で指定されます。) *SIGNON 値は、サブシステムの開始時にワークステーションでサインオン画面が表示されることを示します。 この要件により、サブシステムはシステムおよびサブシステム・レベルのコマンド入力用の対話式装置を必ず持つことになります。 システム終了 (ENDSYS) コマンドは、System i ライセンス・プログラムを終了し、制御サブシステムのコンソールで単一セッション (またはサインオン画面) に移ります。 コンソールのワークステーション項目を含まないサブシステム記述は、制御サブシステムとして開始することはできません。

関連タスク

185 ページの『ワークステーション項目の追加』
ワークステーション項目は、ユーザーがサインオンするか、または他のサブシステムから対話式ジョブを転送するときに、ジョブを開始する場合に使用されます。以下の項目をワークステーション項目に指定できます。パラメーター名は括弧で囲んで示します。ワークステーション項目の追加には、文字ベース・インターフェースを使用します。

190 ページの『ワークステーション項目の変更』
文字ベース・インターフェースを使用して、事前に定義されているワークステーション項目に対して、別のジョブ記述を指定することができます。

194 ページの『ワークステーション項目の除去』
文字ベース・インターフェースを使用して、ワークステーション項目をサブシステム記述から除去できます。サブシステムは、このコマンドの実行時にアクティブ状態にしておくことができます。ただし、ワークステーション項目を通じてアクティブ状態のすべてのジョブは、除去する前に終了しなければなりません。

経路指定項目:

経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目は、サブシステム記述に保管されます。

経路指定項目は、ショッピング・モール・ディレクトリー内の単一の項目にリンクできます。必要なストアを見つけられない顧客は、正しい方向に進むためにディレクトリーを使用できます。同じことはご使用のシステムにも当てはまります。経路指定項目は、ジョブを正しい場所へと導きます。サブシステム記述内の経路指定項目は、呼び出すプログラムを指定し、サブシステム内で実行するジョブの経路指定ステップ、そのジョブが使用するメモリー・プール、および実行時属性の入手元とするクラスを制御します。経路指定データは、使用するジョブの経路指定項目を示します。同時に、経路指定項目および経路指定データは、サブシステム内でのジョブの開始についての情報を提供します。

経路指定項目は、サブシステム記述、クラス、比較データ、最大アクティブ経路指定ステップ、メモリー・プール ID、呼び出すプログラム、スレッド・リソース類縁性、リソース類縁性グループ、および順序番号で構成されます。

関連タスク

185 ページの『経路指定項目の追加』
それぞれの経路指定項目は、ジョブの経路指定ステップを開始するために使用するパラメーターを指定します。経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目をサブシステム記述に追加するには、文字ベース・インターフェースを使用します。

190 ページの『経路指定項目の変更』
指定されたサブシステム記述内の経路指定項目は、文字ベース・インターフェースを使用して変更できます。経路指定項目は、ジョブの経路指定ステップを開始するために使用するパラメーターを指定します。変更を加えるときに、関連サブシステムはアクティブにしておくことができます。

193 ページの『経路指定項目の除去』
文字ベース・インターフェースを使用して、指定されたサブシステム記述から経路指定項目を除去できます。サブシステムは、このコマンドの実行時にアクティブ状態にしておくことができます。ただし、経路指定項目を使用して開始された現在アクティブ状態のジョブがある場合は、その項目を除去することはできません。

クラス:

ジョブ・ランタイム属性は、経路指定項目の (CLS) パラメーターに指定されているクラス・オブジェクトに含まれています。ジョブが複数の経路指定ステップで構成されている場合、後続のそれぞれの経路指定ステップで使用されるクラスは、経路指定ステップを開始するために使用される経路指定項目の中で指定されます。経路指定項目の追加時にクラスが存在しない場合は、修飾クラス名がサブシステム記述に保持されるため、ライブラリー修飾子を指定する必要があります。

経路指定項目クラスに組み込まれているランタイム属性は、以下のとおりです。

実行優先順位 (RUNPTY)

実行優先順位は、1 (最も高い優先順位) から 99 (最も低い優先順位) までの値です。これは、そのジョブが同時にアクティブ状態になっている他のジョブと比べて、処理装置に関して競争する際の優先順位を表します。マルチスレッド・ジョブの場合、実行優先順位は、ジョブ内のどのスレッドに対しても許容される最も高い実行優先順位でもあります。ジョブ内の個々のスレッドの優先順位は、さらに低い場合があります。

タイム・スライス (TIMESLICE)
これは、ジョブ内のスレッドがある程度まとまった量の処理を実行するのに必要な時間を設定するタイム・スライスです。タイム・スライスの終了時に、他のスレッドが記憶域プールでアクティブ状態になるよう、スレッドが非アクティブ状態にされる場合があります。

デフォルトの待ち時間 (DFTWAIT)
これは、ジョブ内のスレッドが、リソースを獲得するためにシステム命令 (LOCK マシン・インターフェース (MI) 命令など) を待つデフォルトの最大時間 (秒単位) を指定します。このデフォルトの待ち時間は、特定の状況で待ち時間がこれ以外の方法で指定されていない場合に使用されます。通常、これは、要求が終了する前にシステム・ユーザーが許容できる待ち時間です。いずれかの命令の待ち時間が超過すると、エラー・メッセージを表示することも、メッセージ・モニター (MONMSG) コマンドで自動的にエラーを処理することもできます。

最大 CPU 時間 (CPUTIME)
これは、ジョブが使用できる最大処理装置時間 (ミリ秒単位) を指定します。ジョブが複数の経路指定ステップで構成されている場合は、各経路指定ステップがこの処理装置時間を使用できます。最大時間が超過すると、ジョブは終了します。

最大一時記憶 (MAXTMPSTG)
これは、ジョブが使用できる最大一時 (補助) 記憶を指定します。ジョブが複数の経路指定ステップで構成されている場合、これは経路指定ステップが使用できる最大一時記憶です。この一時記憶は、プログラム自体で必要な記憶と、ジョブをサポートするために暗黙的に作成される内部システム・オブジェクトで必要な記憶に使用されます。QTEMP ライブラリー内の記憶は含まれません。最大一時記憶が超過すると、ジョブは終了します。このパラメーターは、永続記憶を使用する場合は適用されません。永続記憶はユーザー・プロファイルによって制御されます。

最大スレッド数 (MAXTHD)
これは、このクラスを使用するジョブが任意の時点で実行できるスレッドの最大数を指定します。複数のスレッドが同時に開始された場合、この値を超える可能性があります。この最大数を超過しても、超過したスレッドは通常どおり完了するまで実行されます。追加のスレッドの開始は、ジョブ内の最大スレッド数がこの最大値を下回るまで禁止されます。

テキスト記述 (TEXT)
これは、オブジェクトについて簡潔に記述するテキストを指定します。これはクラス・オブジェクトを作成するときのクラス・オブジェクトの属性で、ジョブのランタイム属性ではありません。

権限 (AUT)
これは、オブジェクトに対する特定権限がないユーザー、権限リストに記載されていないユーザー、およびグループ・プロファイルまたは補足的なグループ・プロファイルにオブジェクトに対する特定権限がないユーザーに付与する権限を指定します。これはクラス・オブジェクトを作成するときのクラス・オブジェクトの属性で、ジョブのランタイム属性ではありません。

比較データ:

経路指定項目の比較値 (CMPVAL) パラメーターは、使用する経路指定項目を判別するために、経路指定データと比較されるデータを指定します。 (経路指定項目は、比較を開始する位置も指定します。)経路指定データは、一致が見つかるまで、各経路指定項目の比較値と順々に比較されます。経路指定項目に含まれる順序番号は、経路指定項目がスキャンされる順序を定義し、経路指定項目の ID として使用することができます。

経路指定データと一致する比較値を持つ経路指定項目が見つかると、経路指定ステップが開始され、経路指定項目に指定されたプログラムが呼び出されます。経路指定項目に関連付けられたクラスのランタイム属性が経路指定ステップで使用され、経路指定項目で指定された記憶域プールで経路指定ステップが実行されます。

比較値 *ANY は、最も大きい順序番号の経路指定項目で指定できます。*ANY は、経路指定データに関係なく一致が強制されることを意味します。比較値 *ANY を含むことができるのは 1 つの経路指定項目だけであり、それはサブシステム記述の最後 (最も大きい順序番号) の項目でなければなりません。

最大アクティブ経路指定ステップ数:

経路指定項目の最大アクティブ経路指定ステップ数 (MAXACT) パラメーターは、この経路指定項目を通じて同時にアクティブにできる経路指定ステップ (ジョブ) の最大数を指定します。

ジョブ内では、同時にアクティブにできる経路指定ステップは 1 つだけです。サブシステムがアクティブになっていて、経路指定ステップの最大数に達すると、その後、この経路指定項目を通じて経路指定ステップを開始しようとしても失敗します。経路指定ステップを開始しようとしたジョブが終了し、メッセージがサブシステムからジョブのログに送信されます。

通常、経路指定ステップ数を制御する理由はないため、推奨値は *NOMAX です。

メモリー・プール ID:

経路指定項目のメモリー・プール ID (POOLID) パラメーターは、プログラムが実行される記憶域プールのプール ID を指定します。ここで指定されるプール ID は、サブシステム記述の記憶域プールと関連があります。

呼び出し対象プログラム

経路指定項目の呼び出し対象プログラム (PGM) パラメーターは、経路指定ステップの最初のプログラム実行として呼び出されるプログラムの名前およびライブラリーを指定します。指定したプログラムにパラメーターを渡すことはできません。プログラム名は経路指定項目に明示的に指定することも、経路指定データから取り出すこともできます。

プログラム名を経路指定項目に指定する場合は、その経路指定項目を選択すると、経路指定項目プログラムが (EVOKE 関数で渡されるプログラム名に関係なく) 呼び出されます。EVOKE 関数に指定したプログラムを呼び出す場合は、このパラメーターに *RTGDTA を指定する必要があります。経路指定項目の追加または変更時にプログラムが存在しない場合は、修飾プログラム名がサブシステム記述に保持されるため、ライブラリー修飾子を指定する必要があります。

順序番号

経路指定項目の順序番号 (SEQNBR) パラメーターは、経路指定データの一致を検索する経路指定項目の順序をサブシステムに通知します。経路指定項目は、順序番号の順に検索されます。経路指定項目をサブシステム記述に追加する場合は、最も頻繁に比較される可能性の高い項目が最初になるように順番を整える必要があります。このようにして検索時間を短縮できます。

順序番号	比較値
10	'ABC'
20	'AB'
30	'A'

順序番号	比較値
40	'E'
50	'D'

上記の例では、経路指定項目は順序番号の順に検索されます。経路指定データが 'A'の場合、検索は経路指定項目 30 で終了します。経路指定データが 'AB'の場合、検索は経路指定項目 20 で終了します。経路指定データが 'ABC'の場合、検索は経路指定項目 10 で終了します。経路指定データは経路指定項目の比較値より長い場合があるため、比較 (左から右の順) は比較値の最後に達した時点で停止します。したがって、経路指定データが 'ABCD'の場合、検索は経路指定項目 10 で終了します。

経路指定項目を定義する場合は、最も具体的なものから最も一般的なものという順序にする必要があります。以下の例は、経路指定項目の正しい定義方法と間違った定義方法を示しています。

正しい方法		間違った方法	
順序番号	比較値	順序番号	比較値
10	'ABC'	10	'ABC'
20	'AB'	20	'ABCD'
30	'A'		
40	'E'		
9999	*ANY		

間違った方法の例では、経路指定項目 20 の比較値と一致する経路指定データは、最初に経路指定項目 10 と一致するため、経路指定項目 20 と一致することはありません。この状態の原因となる比較値で経路指定項目を変更されるか、この状態の原因となる比較値をサブシステム記述に追加すると、システムはこの状態を識別する診断メッセージを送信します。

ジョブの経路指定ステップが開始されると、経路指定項目で指定されるプログラムに制御が渡されます。ジョブの経路指定ステップのランタイム環境を制御するパラメーター (優先順位、タイム・スライスなど) には、経路指定項目に指定したクラスのパラメーターが使用されます。

サブシステムはどのように開始するか

サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

項目がどのように割り振られるかを判別するために、サブシステム記述が使用されます。次のリストは、サブシステムが開始したときに生じるイベントを順番に表します。

1. **サブシステム開始要求が発行される。** サブシステム開始 (STRSBS) コマンドが発行されます。 重要な開始情報は、サブシステム記述にあります。
2. **メモリー・プールが割り振られる。**メモリーはサブシステム記述に定義されたプールに割り振られます。各定義済みプールに割り振られるメモリーは、基本メモリー・プールから取られます。基本記憶域プールに使用できるメモリーの量が、基本メモリー・プール最小サイズ・システム値 QBASPOOL より小さい場合は、システムはプールに記憶域を割り振りません。システムが必要なメモリーすべてを割り振ることができない場合には、システムはその時点で使用可能なメモリーを割り振り、残りはメモリーが使用可能になった時点で割り振ります。
3. **事前開始ジョブが開始する。**事前開始ジョブ項目の情報が使用されます。
4. **自動開始ジョブが開始する。**自動開始ジョブ項目の情報が使用されます。
5. **ディスプレイ装置が割り振られる (サインオン画面が表示される)。** ワークステーション項目があり、装置がオンに構成変更されていて、しかもまだ他のサブシステムによって割り振られていない場合に

は、サブシステムはその装置を割り振り、サインオン画面を表示することができます。 装置がオンに構成変更されていて、しかもすでに他のサブシステムによって割り振られ、現在サインオン画面が表示されている (そのサインオン画面は、 2 番目のサブシステムが開始される前に表示されていた）場合には、 2 番目のサブシステムは最初のサブシステムからその装置を割り振り、サインオン画面を表示できます。 装置がオンに構成変更されていない場合、サブシステムはその装置を割り振ることができません。システム・アービター (QSYSARB) および QCMNARB ジョブは、オフに構成変更されているすべての装置に対してロックのままです。ワークステーション項目は、割り振りを検査する装置に関する情報を提供します。

注: 仮想ディスプレイ装置の場合、装置が完全にオンに構成変更されるとサインオン画面が表示されます。これは、ユーザーがその装置記述を使用して System i に接続する場合に発生します (サインオン画面処理をバイパスするために使用されるデータが接続要求に含まれていないと想定)。以前に作成された装置記述のプールの装置を使用してその接続処理の一部としてオンに構成変更することも、装置を作成してオンに構成変更することもできます。サブシステム開始時に、サブシステムは以前に作成された任意の装置記述に対してロックを保留します。

6. **ジョブ待ち行列が割り振られる。**サブシステムがすでに別のアクティブ・サブシステムに割り振られている場合、サブシステムはジョブ待ち行列を割り振ることはできません。ジョブ待ち行列項目の情報が使用されます。
7. **通信装置が割り振られる。**要求は、QLUS (LU サービス) システム・ジョブに送られます。このシステム・ジョブは、すべての通信装置の装置割り振りを処理します。通信項目の情報が使用されます。
8. **環境が作業可能になる。**

関連タスク

181 ページの『サブシステムの開始』
サブシステム開始 (STRSBS) コマンドは、コマンドに指定されたサブシステム記述を使用してサブシステムを開始します。 サブシステムが開始されると、システムは、サブシステム記述に指定された必要で、しかも使用可能なリソース（記憶域、ワークステーション、およびジョブ待ち行列）を割り振ります。サブシステムを開始するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

ワークステーション装置はどのように割り振られるか:

サブシステムは、AT(*SIGNON) ワークステーション項目のサブシステム記述内のすべてのワークステーション装置を割り振ろうとします。

サブシステムの開始時に、以下の状況が生じる可能性があります。

- 装置がオンに構成変更されていない場合、サブシステムはその装置を割り振ることができません。システム・アービター (QSYSARB) および QCMNARBxx ジョブは、オフに構成変更されているすべての装置に対してロックのままです。
- 装置がオンに構成変更されていて、しかもまだ他のサブシステムによって割り振られていない場合には、サブシステムはその装置を割り振り、サインオン画面を表示することができます。
- 装置がオンに構成変更されていて、しかもすでに他のサブシステムによって割り振られ、現在サインオン画面が表示されている (そのサインオン画面は、 2 番目のサブシステムが開始される前に表示されていた）場合には、 2 番目のサブシステムは最初のサブシステムからその装置を割り振り、サインオン画面を表示できます。

複数のサブシステムが同じワークステーション (ワークステーション項目に指定されているのと同じもの) を割り振ろうとする場合で、ワークステーションがオフに構成変更されている場合は、ワークステーションがオンに構成変更されたときにどのサブシステムがワークステーションを取得するかを予測することはでき

ません。同様に、ワークステーション項目が、ワークステーション名ではなくワークステーション・タイプを指定する場合、サブシステムはそのタイプのワークステーションのすべてまたは一部を取得するか、1 つも取得しない可能性もあります。(これは、総称名を持つワークステーション項目にも当てはまります。)このような状況を避けるために、複数のサブシステムが同じワークステーションを使用することがないよう、サブシステムのワークステーション項目をセットアップできます。

ユーザーのサインオン後

ユーザーがワークステーションにサインオンすると、ワークステーション上のサインオン画面に表示されたサブシステム内でジョブが実行されます (サブシステムは IBM 提供のサインオン画面で確認できます)。ユーザーのサインオン後に、以下の状況が生じる可能性があります。

- 2 番目のサブシステムが開始され、ユーザーがサインオンしたワークステーションを割り振ろうとする場合、2 番目のサブシステムはそのワークステーションを割り振ることはできません。ユーザーのジョブは、引き続き最初のサブシステムで実行されます。
- ユーザーがシステム要求メニューでオプション 1 (代替ジョブのサインオンの表示 (Display signon for alternative job)) を選択するか、2 次ジョブへの移行 (TFRSECJOB) コマンドを発行すると、新しいジョブは元のジョブと同じサブシステムで実行されます。
- ユーザーがサインオフすると、ジョブ転送 (TFRJOB) コマンドを使用してユーザーがサブシステムに転送され、このワークステーションのワークステーション項目に AT (*ENTER) が指定されない限り、ワークステーションはユーザーがサインオンしたときに使用されたサブシステムに割り振られたままになります。 サインオン画面が表示され、そのワークステーションの後続のすべてのジョブは継続してそのサブシステムで実行されます (サインオン画面に表示されている間にワークステーションを割り振る別のサブシステムが開始されない限り)。
- ユーザーがサインオフし、ユーザーのジョブが実行されていたサブシステムが終了すると、装置は割り振り解除されます。次いで、2 番目のサブシステムが、装置を割り振り、サインオン画面を表示できるようになります。

  関連タスク

  特定のサブシステムへのユーザーの割り当て
  いくつかの手法を用いて、装置名を割り当てた後、その装置名をユーザーと関連付けることができます。これが完了したら、ワークステーション項目を使用して、ユーザーを正しいサブシステムに割り当てることができます。

  197 ページの『特定のサブシステムへのユーザーの割り当て』
  いくつかの手法を用いて、装置名を割り当てた後、その装置名をユーザーと関連付けることができます。これが完了したら、ワークステーション項目を使用して、ユーザーを正しいサブシステムに割り当てることができます。

  関連情報

  経験報告: サブシステム構成

  Telnet 出口点プログラムの使用

シナリオ: ワークステーションの割り振り:

この例では、2 つのワークステーションがどのように 2 つの別々のサブシステムに割り振られるかを示します。

このシナリオでは、サブシステム A およびサブシステム B のサブシステム記述にワークステーション DSP01 および DSP02 があります (ワークステーション項目は AT(*SIGNON)) を指定)。

装置名	割り振り先
DSP01	サブシステム A
DSP02	サブシステム A

サブシステム A の開始時に、両方のワークステーションがオンに構成変更されるとします。

サブシステム A は、両方のワークステーションを割り振り、両方のワークステーションにサインオン画面を表示します。 サブシステム A でサインオン画面がワークステーションに表示されていますが、別のシステムまたはジョブがこれらのサブシステムを割り振ることが可能です。その場合、サブシステム A はそのワークステーションを使用できなくなります。

装置名	割り振り先
DSP01	USER1
DSP02	サブシステム A

ユーザー (USER1) がワークステーション DSP01 にサインオンすると、装置が USER1 のジョブ (サブシステム A で実行中) に割り振られます。ワークステーション DSP02 では、サインオン画面が表示されたままです。 したがって、別のサブシステムまたはジョブによって割り振り可能です。その場合、サブシステム A はそのワークステーションを使用できなくなります。

装置名	割り振り先
DSP01	USER1
DSP02	サブシステム B

サブシステム B が開始されます。USER1 がワークステーション DSP01 にサインオンしているため、サブシステム B は装置を割り振ることができません。サブシステム B は、装置が使用可能になると装置の割り振りを要求します。DSP02 は、サブシステム A でだれもサインオンしていないため、サブシステム B に割り振られます。DSP02 で開始したジョブは、サブシステム B で実行されます。

装置名	割り振り先
DSP01	サブシステム A
DSP02	サブシステム B

USER1 がサインオフします。ユーザー・ジョブがサブシステム A で実行されていたため、別のユーザーがワークステーションにサインオンしてサブシステム A で実行できるよう、そのサブシステムにサインオン画面が表示されます。サブシステム A が終了すると、ワークステーション DSP01 がサブシステム B によって割り振られます (装置を割り振る未処理の要求があるため)。

現在ワークステーションが割り振られているサブシステムの名前が、IBM 提供のサインオン画面の右上隅に表示されます。

関連タスク

197 ページの『特定のサブシステムへのユーザーの割り当て』
いくつかの手法を用いて、装置名を割り当てた後、その装置名をユーザーと関連付けることができます。これが完了したら、ワークステーション項目を使用して、ユーザーを正しいサブシステムに割り当てることができます。

関連情報

Telnet 出口点プログラムの使用

メモリー・プール

メモリー・プールとは、メイン・メモリーまたは記憶域の論理的な区画のことで、ジョブまたはジョブ・グループの処理のために予約されています。システム上のすべての主記憶域はメモリー・プールと呼ばれる論理的な割り振りに区分されます。デフォルトでは、システムはメモリー・プールへのデータおよびプログラムの転送も管理します。

ジョブがメモリーを取得するメモリー・プールは常に同じプールであり、アクティビティー・レベルを制限しています。 (メモリー・プールのアクティビティー・レベルは、メモリー・プール内で同時にアクティブ状態にできるスレッドの数です。) この例外として、システム・ジョブ (Scpf、Qsysarb、および Qlus など) は、基本プールからメモリーを取得しますが、マシン・プール・アクティビティー・レベルを使用します。さらに、サブシステム・モニターはそのメモリーをサブシステム記述の最初のプールから入手しますが、マシン・プールのアクティビティー・レベルを使用します。したがって、サブシステム・モニターは、アクティビティー・レベルの設定には関係なく、いつでも実行が可能です。

メモリー・プールを使用する理由

プールの数やサイズを制御することによって、サブシステムで実行できる作業の量を制御することができます。サブシステムに割り当てられるプールのサイズが大きくなればなるほど、より多くの作業をそのサブシステムで行うことが可能になります。

共用記憶域プールを使用することによって、システムは対話式ユーザー用のジョブを複数のサブシステムに分散することができ、しかもそれらのジョブを同じ記憶域プールで実行することができます。

サブシステム内の複数のプールは、システム・リソースにおけるジョブの競合を制御するために役立ちます。サブシステム内に複数のプールを持つことの利点は、実行される作業量と、それらのジョブの応答時間を分割できるということです。たとえば、日中に応答時間が速い対話式ジョブを実行したいとします。効率を良くするために、対話式プールをさらに大きくすることができます。夜間には数多くのバッチ・ジョブを実行するために、バッチ・プールをさらに大きくします。

注: システムのチューニングや管理は、システムでのワークフローの効率を高めるのに役立ちますが、不適切なハードウェア・リソースには対処できません。作業負荷の所要量が著しく大きい場合、ハードウェアのアップグレードを考慮してください。

メモリー・プール内でのデータの処理方法

データが既に主記憶域にある場合、それが入っているメモリー・プールに依存せずに参照できます。ただし、必要とされるデータがどのメモリー・プール内にも存在しない場合、それを参照するジョブと同じメモリー・プール内にそれを入れます (これはページ不在として知られています)。データを転送してメモリー・プールに入れると、他のデータは取り除かれ、変更すると、補助記憶域に自動的に記録されます (これはページングと呼ばれます)。メモリー・プールのサイズは、転送率はパフォーマンスに影響を与えるので、妥当なレベルでのデータ転送 (ページング) を継続できる十分な大きさにする必要があります。

関連概念

200 ページの『メモリー・プールの管理』
ジョブが効率的に完了するためメモリーが十分あるようにするのは重要なことです。サブシステム A にメモリーが過剰に与えられて、サブシステム B ではメモリーが不足する場合、サブシステム B のジョブは効率よく実行されない可能性があります。以下の情報では、メモリー・プールの管理に関連した種々のタスクについて説明されています。

関連情報

Retrieve System Status (QWCRSSTS) API

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

メモリー・プールの種類

システム上のすべての主記憶域はメモリー・プール と呼ばれる論理的な割り振りに区分されます。システムのすべてのメモリー・プールは、私用または共用のいずれかです。専用メモリー・プール、共用メモリー・プール、および特殊共用メモリー・プールがあります。私用プールと共用プールを合わせて最大 64 個のメモリー・プールまで、同時にアクティブ状態にすることができます。

私用メモリー・プール

私用メモリー・プール (ユーザー定義メモリー・プールとも呼ばれる) には、単一のサブシステムがジョブを実行するために使用できる特定量の主記憶域が入っています。これらのプールは、複数のサブシステムで共用することはできません。これらは、System i ナビゲーター によってサブシステム名で識別されます。62 個まで私用メモリー・プールを割り振って、アクティブ・サブシステムで使用することができます。

共用メモリー・プール

共用プール は、特殊または汎用のいずれかです。マシン・プールおよび基本プールは特殊共用プールと見なされ、それ以外のすべてのプールは汎用共用プールと見なされます。システムで定義されている共用メモリー・プールは 64 個あり、そのうちの 63 個についてはサブシステム記述の作成時点でその使用を指定することができます (マシン・プールはシステム用に予約されています)。

特殊共用プール (*MACHINE および *BASE)

***MACHINE**

マシン・メモリー・プールは、共用されたマシンおよびオペレーティング・システムで使用されます。System i ナビゲーター では、マシンとして識別されます。マシン・メモリー・プールは、ユーザーの介入を必要とせずにシステムが実行するタスクに対して記憶域を提供します。このメモリー・プールのサイズは、マシン・メモリー・プール・サイズのシステム値 (QMCHPOOL) で指定されます。 このメモリー・プールではユーザー・ジョブは実行されません。(システム状況処理 (WRKSYSSTS) 画面で、マシン・メモリー・プールはシステム・プール ID 1 として表示されます。)

***BASE**

基本メモリー・プールは、System i ナビゲーターでは基本として示されます。これにはシステム上の割り当てられていないすべての主記憶域 (つまり、他のメモリー・プールが必要としていないすべての主記憶域) が入っています。基本プールには、多くのサブシステムで共用できる記憶域が含まれます。基本メモリー・プールは、バッチ処理や種々のシステム機能で使用されます。基本メモリー・プールの最小サイズ (QBASPOOL) システム値は、基本メモリー・プールの最小サイズを指定します。このメモリー・プールのアクティビティー・レベルは、基本メモリー・プール用の最大適格スレッド数 (QBASACTLVL) システム値で指定されます。 (システム状況処理 (WRKSYSSTS) 画面で、基本メモリー・プールはシステム・プール ID 2 として表示されます。)

汎用共用プール

汎用共用プール は、複数のサブシステムが同時に使用できる主記憶域のプールです。文字ベース・インターフェースでは、以下のように識別されます。

- *INTERACT。対話式ジョブに使用される対話式記憶域プール。
- *SPOOL。スプール書き出しプログラムに使用される記憶域プール。
- *SHRPOOL1 から *SHRPOOL60。自由に使用できる記憶域プール。

System i ナビゲーターでは、汎用共用プールは、対話式、スプール、および共用 1 から 共用 60 と識別されます。

関連タスク

207 ページの『専用メモリー・プールの作成』
専用メモリー・プール (ユーザー定義メモリー・プールとしても知られる) は、IBM 提供のサブシステムまたはユーザー定義サブシステムで使用できます。サブシステムは最大で 10 のメモリー・プール定義を持つことができます。専用メモリー・プールはサブシステム記述内に作成します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

パフォーマンス・システム値: マシン記憶域プール・サイズ

パフォーマンス・システム値: 基本記憶域プールの最小サイズ

パフォーマンス・システム値: 基本記憶域プール有資格スレッドの最大数

プール番号付けのスキーム

プールには 2 つの番号付けスキームがあり、1 つはサブシステム内で使用し、もう 1 つはシステム全体で使用します。サブシステムは、使用するプールを参照する一連の番号を用います。ですから、サブシステム記述を作成または変更する場合、1 つ以上のプールを定義し、それらに 1、2、3 のようにラベル付けを行えます。これはサブシステム・プールの指定であって、システム状況処理 (WRKSYSSTS) で表示されるプール番号には対応していません。

システム上のすべてのプールの状況を追跡するには、異なる番号セットを使用します。サブシステム処理 (WRKSBS) 画面は、サブシステム・プール ID および列見出しをシステム・プール ID と関連付けます。

```
                          サブシステムの処理
                                                  システム: XXXXXXXX
オプションを入力して、実行キーを押してください。
  4= サブシステムの終了 5= サブシステム記述の表示
  8= サブシステム・ジョブの処理

                     合計     -------- サブシステム・プール ---------
OPT サブシステム   記憶域 (M)  1   2   3   4   5   6   7   8   9   10
 _  NYSBS             .48     2   4   5
 _  PASBS             .97     2   6   5
 _  QINTER          11.71     2   3




                                                            終わり
パラメーターまたはコマンド===>
F3= 終了   F5= 最新表示   F11= システム・データの表示   F12= 取り消し
F14= システム状況の処理
```

例: プールが番号付けされる方法

以下の例は、プールが番号付けされる方法を示しています。

サブシステム		
CRTSBSD QINTER プール (1 *基本) (2 1200 25) (システム・プール 2、3)	CRTSBSD NYSBS プール (1 *基本) (2 500 3) (3 *SHRPOOL2) (システム・プール 2、4、5)	CRTSBSD PASBS プール (1 *基本) (2 1000 3) (3 *SHRPOOL2) (システム・プール 2、5、6)

QINTER の開始後、以下のプールが割り振られます。

システム・プール番号	説明	QINTER
1	*マシン・プール	
2	*基本プール	1
3	QINTER 私用プール	2

NYSBS の開始後、以下のプールが割り振られます。

システム・プール番号	説明	QINTER	NYSBS
1	*マシン・プール		
2	*基本プール	1	1
3	QINTER 私用プール	2	
4	NYSBS 私用プール		2
5	*SHRPOOL2 共用プール		3

PASBS の開始後、以下のプールが割り振られます。

システム・プール番号	説明	QINTER	NYSBS	PASBS
1	*マシン・プール			
2	*基本プール	1	1	1
3	QINTER 私用プール	2		
4	NYSBS 私用プール		2	
5	SHRPOOL2 共用プール		3	3
6	PASBS 私用プール			2

関連タスク

204 ページの『共用プールのチューニング・パラメーターの管理』
共用プールのチューニング・パラメーターを管理するには、System i ナビゲーター または文字ベース・インターフェースのコマンドを使用します。

204 ページの『プールの構成の管理』
プール・サイズ、アクティビティー・レベル、またはページング・オプションを変更するには、 System i ナビゲーター または文字ベース・インターフェースのコマンドを使用します。

205 ページの『メモリー・プール・サイズの変更』
メモリー・プールのサイズ変更は、サブシステムが処理できる作業の量に直接影響します。多くのメモリーが与えられれば、サブシステムはさらに多くの作業を行えるようになります。メモリー・プールのパラメーターの変更を開始する前には、システムを注意深くモニターすることが大切です。さらに、再調整が必要になる場合もあるため、調整後もこれらのレベルを定期的に確認してください。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

メモリー・プールの割り振り

サブシステムを開始すると、システムは、開始されたサブシステムのサブシステム記述で定義されたユーザー定義の記憶域プールを割り振ろうとします。

システムが必要な記憶域すべてを割り振ることができない場合には、その時点で使用可能な限りの記憶域を割り振り、残りの記憶域が使用可能になった時点で割り振ります。たとえば、以下の表について考慮してください。700KB が使用可能で、*SHRPOOL2 が 500KB に定義されている場合、最初の記憶域プールには 300KB が割り振られ、2 番目の記憶域プールには 400KB が割り振られます。

SBSD で指定のプール ID	1	2
必要な記憶域	300K	*SHRPOOL2
システム・プール ID	3	4
割り振られる記憶域	300K	400K
アクティビティー・レベル	1	
プール・タイプ	専用	共用

定義した記憶域プールが割り振られると、基本メモリー・プールのサイズは減少します。私用プールは基本メモリー・プール内で使用可能なため、システムは私用プールのみに可能な限り多くの記憶域を割り振ります。基本メモリー・プールの最小サイズ (QBASPOOL) システム値によって、最小の基本プール・サイズが決まります。

関連タスク

200 ページの『メモリー・プール情報の表示』
System i ナビゲーター または文字ベース・インターフェースを使用して、システム上にあるメモリー・プールについての情報を表示できます。

201 ページの『メモリー・プールを使用したサブシステム数の判別』
複数のジョブを実行するために、サブシステムには特定の比率でメモリーが割り振られます。いくつの異なるサブシステムが同じメモリー・プールからプルしているかを知っていることは重要です。いくつのサブシステムがプールにジョブを送り出しているか、またいくつのジョブがプールで実行されているかを知ると、プールのサイズとアクティビティー・レベルを調整して、リソースの競合を減らしたいと場合もあります。

202 ページの『メモリー・プール内のジョブ数の判別』
System i ナビゲーター には、メモリー・プール内で現在実行中のジョブのリストを素早く表示する手段が備えられています。

203 ページの『単一ジョブを実行しているプールの判別』
期待通りに実行できないジョブがある場合、そのジョブを実行中のメモリー・プールを確認することもできます。単一ジョブが実行されているプールを判別するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

メモリー・プールのアクティビティー・レベル

メモリー・プールのアクティビティー・レベルは、メモリー・プール内で CPU を同時に使用できるスレッドの数です。こうすることによって、システム・リソースを効果的に使用できます。アクティビティー・レベルの制御はシステムが管理します。

スレッドの処理過程で、システム・リソースやワークステーション・ユーザーからの応答をプログラムが待機する状態になることがしばしばあります。このような待機の過程では、スレッドはメモリー・プールのアクティビティー・レベルの使用を放棄し、処理可能状態にある他のスレッドがそれに代わります。

同時に実行できるスレッドの数を超える多くのスレッドが開始された場合、過剰なスレッドは処理装置が使用できるまで待機しなければなりません (通常、これは短時間です)。メモリー・プールのアクティビティー・レベルによって、サブシステムの種々のメモリー・プールにおけるメイン・メモリーの競合の度合いを制限することができます。

実行中のスレッド (またはアクティブ・スレッド) の数は、プロセッサーの競合に適していて、メモリー・プールのアクティビティー・レベルに対してカウントされるスレッドの数を指します。したがって、入力、メッセージ、装置の割り振り、またはファイルのオープンを待機しているスレッドは、アクティブ・スレッドには含めません。不適格スレッド (スレッドが実行可能であっても、メモリー・プールのアクティビティー・レベルが最大になっている場合) は、アクティブ・スレッドには含めません。

アクティビティー・レベルはどのように機能するか

スレッドの処理は補助記憶域から必要なデータを取り出すときにわずかに中断されるため、メモリー・プールで同時に複数のスレッドをアクティブ状態にすることができます。この遅延時間は通常は短時間ですが、この間に他のスレッドを実行することができます。アクティビティー・レベルを使用することによってマシンは、1 つのメモリー・プールで多くのスレッドを処理し、同時に、競合のレベルをユーザーが指定する範囲に保つことができます。

最大アクティビティー・レベル

メモリー・プールの最大アクティビティー・レベルに達すると、メモリー・プールを必要とする追加のスレッドは不適格状態になり、メモリー・プールのアクティブ・スレッドの数が最大アクティビティー・レベルより低くなるか、あるいはスレッドがそのタイム・スライスの終了に至るまで待ちます。あるスレッドがメモリー・プールの使用を放棄すると、アクティブ状態でない別のスレッドがその優先順位にしたがって実行適格になります。たとえば、実行中のスレッドがワークステーションからの応答を待機する状態になると、そのスレッドはアクティビティー・レベルを放棄し、アクティビティー・レベルは最大のレベルではなくなります。

メモリー・プールのアクティビティー・レベルの定義

メモリー・プールおよびアクティビティー・レベルが正しく定義されているかどうかは、メモリー・プールのサイズ、CPU の数、ディスク装置アームの数、およびアプリケーションの特性に依存します。

関連タスク

200 ページの『メモリー・プール情報の表示』
System i ナビゲーター または文字ベース・インターフェースを使用して、システム上にあるメモリー・プールについての情報を表示できます。

201 ページの『メモリー・プールを使用したサブシステム数の判別』
複数のジョブを実行するために、サブシステムには特定の比率でメモリーが割り振られます。いくつの異なるサブシステムが同じメモリー・プールからプルしているかを知っていることは重要です。いくつのサブシステムがプールにジョブを送り出しているか、またいくつのジョブがプールで実行されているかを知ると、プールのサイズとアクティビティー・レベルを調整して、リソースの競合を減らしたいと場合もあります。

202 ページの『メモリー・プール内のジョブ数の判別』
System i ナビゲーター には、メモリー・プール内で現在実行中のジョブのリストを素早く表示する手段が備えられています。

203 ページの『単一ジョブを実行しているプールの判別』
期待通りに実行できないジョブがある場合、そのジョブを実行中のメモリー・プールを確認することもできます。単一ジョブが実行されているプールを判別するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

ジョブ

システムで実行されるすべての作業はジョブによって実行されます。それぞれのジョブには、システム内で固有の名前があります。システム・ジョブを除くすべてのジョブは、サブシステムの中で実行されます。ジョブは、ジョブ待ち行列項目、ワークステーション項目、通信項目、自動開始ジョブ項目、または事前開始ジョブ項目などのどの実行処理項目からでもサブシステムに入力できます。

各アクティブ・ジョブには少なくとも 1 つのスレッド (初期スレッド) があり、追加の 2 次スレッドが含まれている場合もあります。スレッドは独立した作業単位です。ジョブ属性はジョブのスレッド間で共有されますが、スレッドは、呼び出しスタックなど独自の属性も有しています。ジョブの属性には、 作業の処理方法に関する情報が含まれます。同一ジョブ内で複数のスレッドに共用される属性に対して、ジョブは所有者としての役割を果たします。実行管理機能によって、ジョブの属性によるシステム上の作業の実行を制御する方法が提供されます。

適切な権限

ジョブの属性に変更を加えるには、ほとんどの場合、ユーザーがジョブ制御特殊権限 (*JOBCTL) を持っているか、ユーザー・プロファイルが、変更するジョブのジョブ・ユーザー ID と一致している必要があります。

*JOBCTL 特殊権限がないと変更できない属性がいくつかあります。以下の属性です。

- デフォルトの待ち時間
- 実行優先順位
- タイム・スライス

注: ジョブのアカウンティング・コードに変更を加える場合、 *JOBCTL 特殊権限またはジョブのジョブ・ユーザー ID に一致するユーザー・プロファイルに加えて、「アカウンティング・コードの変更 (CHGACGCDE)」コマンドに対する *USE 権限が必要です。

ジョブ待ち行列、出力待ち行列、およびソート・シーケンス・テーブルなどの i5/OS オブジェクトを参照するジョブ属性の場合、そのオブジェクトに対する適切な権限が必要です。 i5/OS 権限について詳しくは、「Security Reference」の一連のトピックの『コマンドによって使用されるオブジェクトで必要な権限』を参照してください。

関連概念

37 ページの『ジョブ・ユーザー ID』
ジョブ・ユーザー *ID (JUID)* は、あるジョブが他のジョブによって識別されるユーザー・プロファイルの名前です。この名前は、他のジョブがこのジョブを操作しようとする際の権限許可検査に使用されます。

ジョブの特性

実行管理機能によって、ジョブの属性によるシステム上の作業の実行を制御する方法が提供されます。しかし、ジョブの様々な性質を制御できるようになるには、ジョブの多様な特性を理解する必要があります。

以下の情報は、ジョブの特性について取り上げています。

ジョブ名の構文:

システム上でのジョブの制御や識別を容易にするため、各ジョブには固有の修飾ジョブ名があります。修飾ジョブ名は、ジョブ名 (または単純ジョブ名)、ユーザー名、およびジョブ番号の 3 つで構成されています。

- 対話式ジョブの場合、ジョブ名はサインオンしたワークステーションまたはエミュレーター・セッションの名前と同じです。バッチ・ジョブの場合、独自のジョブ名を指定できます。ジョブ名の長さは最大 10 文字です。
- ユーザー名は、ジョブが開始されるユーザー・プロファイルの名前です。対話式ジョブの場合、ユーザー名はシステムにサインオンするために使用したユーザー・プロファイルの名前です。 これは、サインオン画面の「ユーザー」フィールドに入力したユーザー名です。 Telnet を使用してサインオンをバイパスする場合には、これはシステムに自動的にサインオンするために使用するユーザー名です。 バッチ・ジョブの場合、バッチ・ジョブが実行されるユーザー・プロファイルを指定できます。ユーザー名の長さは最大 10 文字です。
- ジョブ番号はシステムによって割り当てられた固有の番号で、2 つ以上の同じジョブ名およびユーザー名があるとしても、ジョブを識別することができます。ジョブ番号は、必ず 6 桁の数字です。

構文

修飾ジョブ名の構文は、オブジェクトの修飾名に似ています。たとえば、ジョブ名が DSP01 で、ユーザーが QPGMR、さらにジョブ番号が 000578 の場合、修飾ジョブ名はジョブ処理 (WRKJOB) コマンドで次のように入力します。

```
WRKJOB JOB(000578/QPGMR/DSP01)
```

オブジェクト名と似ている別の点として、すべての修飾子を指定する必要はありません。たとえば、以下について考慮してください。

```
WRKJOB JOB(QPGMR/DSP01)
```

または

```
WRKJOB JOB(DSP01)
```

これは、完全修飾ジョブ名を入力したのと同じように作動します。システム上の幾つかのジョブが入力したジョブ名の一部と一致する場合、「Select Job (ジョブの選択)」画面が表示されます。この画面を使用すると、重複するジョブ名のリストからジョブを選択することができます。

ジョブ属性:

ジョブ属性は、システムが各ジョブを実行する方法を決定します。一部のジョブ属性は、ユーザー・プロファイルから設定されます。その他のジョブ属性は、システム値、ロケール、ジョブ投入 (SBMJOB) コマンド、ジョブ記述、およびジョブ変更 (CHGJOB) コマンド (ジョブ実行時に属性の値を変更できる) に由来します。

ジョブ属性を制御すると、ジョブ・レベル、ユーザー・レベル、またはシステム・レベルでジョブを柔軟に制御できるようになります。たとえば、ジョブ属性を取得するためのシステム値 (これがシステム・デフォルト) に至るまでシステムのセットアップを徹底的に行うことができます。その後、システム上のすべての新しいジョブの値を変更したい場合には、システム値を変更できます。

ジョブ記述に値を指定して、そのジョブ記述を使用するすべてのタイプのジョブに影響を及ぼすことができます。たとえば、すべてのバッチ・ジョブが同じジョブ記述を使用している場合、そのバッチ・ジョブのジョブ記述を変更すると、すべてのバッチ・ジョブに影響を及ぼし、その他すべてのジョブには作用しないままにできます。

関連情報

経験報告: 実行管理ジョブ属性

ジョブ記述:

ジョブ記述により、複数のユーザー用に保管されて使用できる、ジョブ属性のセットを作成できます。ジョブ記述は、いくつかのジョブ属性のソースとして使用でき、システムにジョブの実行方法を通知します。属性情報は、ジョブの開始時刻、ジョブの送信元、ジョブの実行方法などの情報をシステムに通知します。ジョブ記述は多くのジョブが使用可能なテンプレートと見なすことができ、それによって各ジョブに設定する必要のある特定のパラメーターの数を減らすことができます。

ジョブ記述は、自動始動、バッチ、対話式、および事前開始の各ジョブ・タイプによって使用されます。複数のジョブに同じジョブ記述を使うことができます。ジョブを定義する際、ジョブ記述を次の 2 つの方法のいずれかで使用できます。

- 属性を指定変更することなく指定のジョブ記述を使用する。たとえば、

  ```
  SBMJOB JOB(OEDAILY) JOBD(QBATCH)
  ```

- 指定のジョブ記述の属性を使用し、その一部の属性を指定変更します (BCHJOB コマンドまたは SBMJOB コマンドを使用)。たとえば、ジョブ記述 QBATCH のメッセージ・ロギングを指定変更するには、次のように指定します。

  ```
  SBMJOB JOB(OEDAILY) JOBD(QBATCH)       LOG(2 20 *SECLVL)
  ```

  注: 自動開始ジョブ、ワークステーション・ジョブ、または通信ジョブのジョブ記述属性を指定変更することはできません。

関連タスク

129 ページの『ジョブ記述の作成』
文字ベース・インターフェース、ジョブ記述処理 (WRKJOBD) コマンド、またはジョブ記述作成 (CRTJOBD) コマンドを使用して、ジョブ記述を作成します。

129 ページの『ジョブ記述の使用』
ジョブ記述の最も一般的な使用法は、ジョブ投入 (SBMJOB) コマンドに指定するという方法です。ジョブ記述 (JOBD) パラメーターに、このジョブで使用するジョブ記述を指定します。 バッチ・ジョブを指定する場合は、以下の 2 つの方法のいずれかによりジョブ記述を使用できます。

ジョブ記述とセキュリティー:

システム内のすべてのジョブは、ジョブ開始時にジョブ記述を使用します。ジョブ記述は、ジョブの様々な属性を制御します。USER パラメーターは、ジョブに割り当てられるユーザー・プロファイルの名前を制御します。指定のユーザー・プロファイル名 (USER) を持つジョブ記述は、特定のユーザーに対してのみ権限が与えられるようにしてください。そのようにしないと、セキュリティー・レベル 30 以下になり、他のユーザーはそのユーザー・プロファイルでジョブを投入して実行できるようになります。

次の例を考慮してください。

```
CRTJOBD JOBD(XX) USER(JONES) . . . AUT(*USE). . AUT(*USE)
```

この例では、任意のユーザーが XX ジョブ記述を使用してジョブを投入でき、JONES に権限が与えられているすべてのことに対して権限が与えられるので、セキュリティー・リスクが生じます。こうしたタイプのジョブ記述がワークステーション項目で使用されると、すべてのユーザーが Enter キーを押すだけでそうしたユーザーとしてサインオンできます。機密漏れを避けるには、このタイプのジョブ記述に *PUBLIC の権限を与えないでください。

注: セキュリティー・レベル 40 および 50 の場合、ジョブ投入 (SBMJOB) コマンドではサブミッターがジョブ記述で指定されたユーザー・プロファイルに対して権限を与えられていなければなりません (*USE)。 この場合、SBMJOB がユーザー (*JOBD) を指定すると想定します。 それでも、特定の理由 (自動開始ジョブなど) によって必要な場合やアクセスを厳重に制御する必要がある場合を除いては、ジョブ記述内でユーザーを指定しないでください。

USER パラメーターおよび対話式ジョブ

使用するジョブ記述は、ワークステーション追加項目 (ADDWSE) コマンドで定義します。 デフォルトでは、ユーザー・プロファイルのジョブ記述を使用します。USER(*RQD) がジョブ記述で指定されると、ユーザーはユーザー名を入力する必要があります。USER(xxxx) が指定される場合 (ここで、xxxx は特定のユーザー・プロファイル名)、セキュリティー・レベルが 40 以上でない限りは、ユーザーはサインオン画面で Enter キーを押して、xxxx ユーザー・プロファイル名で操作をできるようになります。

USER パラメーターおよびバッチ・ジョブ

バッチ・ジョブに使用するジョブ記述は、ジョブ投入 (SBMJOB) コマンドまたはバッチ・ジョブ (BCHJOB) コマンドで指定されます。

BCHJOB コマンドが含まれる入力ストリームが入れられる場合、読み取りプログラム開始コマンド (STRDBRDR、STRDKTRDR) のいずれか、またはジョブ投入コマンド (SBMDBJOB、SBMDKTJOB など) の 1 つを入力するユーザーは、指定されるジョブ記述に対してオブジェクト操作権限 (*OBJOPR) を持っていなければなりません。入力ストリームが使用されると、ジョブ待ち行列にジョブを配置しているユーザ

ーのユーザー・プロファイルではなく、ジョブ記述のユーザー・プロファイルでジョブが必ず操作されます。ジョブ記述で USR(*RQD) が指定される場合には、BCHJOB コマンド上のジョブ記述の使用は無効になります。

SBMJOB コマンドが使用されると、このコマンドがデフォルトになり、サブミッターのユーザー・プロファイル名でバッチ・ジョブは操作されます。 しかし、 USER(*JOBD) が SBMJOB コマンドで指定されると、ジョブ記述の USER パラメーターで指定される名前でジョブは操作されます。

ユーザーが特定のユーザー・プロファイルの作業を実行するには、しばしばジョブ記述内の特定の名前が必要となります。たとえば、そのために QBATCH ジョブ記述が USER(QPGMR) と共に使われます。機密漏れを避けるには、このタイプのジョブ記述に *PUBLIC の権限を与えないでください。

呼び出しスタック:

呼び出しスタック は、ジョブに対して現在実行しているすべてのプログラムまたはプロシージャーの番号付きリストです。プログラムおよびプロシージャーは、CALL 命令によって明示的に開始することも、他の何らかのイベントによって暗黙的に開始することもできます。

呼び出しスタックは、ジョブ・レベルとスレッド・レベルの両方で使用可能です。文字ベース・インターフェース上では、呼び出しスタックは、呼び出しスタック項目の後入れ先出し法 (LIFO) リストであり、呼び出されるプロシージャーまたはプログラムごとに 1 つの項目があります。 System i ナビゲーター では、デフォルトではスタック内の最後の項目がリストの先頭に表示されます。 ただし、順序は「**昇順にソート (Sort ascending)**」または「**降順にソート (Sort descending)**」ボタンを使用して変更できます。

「呼び出しスタック (Call Stack)」画面に組み込まれている情報には、オリジナル・プログラム・モデル (OPM)、統合言語環境 (ILE)、i5/OS ポータブル・アプリケーション・ソリューション環境 (PASE)、および Java™ アプリケーションの呼び出し情報が含まれています。 さらに、*SERVICE 特殊権限があるユーザー・プロファイル下で実行中の場合、ライセンス内部コード (LIC) および i5/OS PASE カーネル用の追加の項目を表示できます。

関連タスク

124 ページの『呼び出しスタックの表示』
ジョブまたはスレッドの呼び出しスタックを表示するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

クラス・オブジェクト:

クラス・オブジェクトには、ジョブのランタイム環境を制御する実行属性が含まれています。 IBM 提供のクラス・オブジェクトまたはクラスは、標準的な対話式アプリケーションおよびバッチ・アプリケーションの両方の必要を満たします。以下のクラス (名前別) がシステムに提供されます。

- QGPL/QBATCH: バッチ・ジョブで使用
- QSYS/QCTL: 制御サブシステムで使用
- QGPL/QINTER: 対話式ジョブで使用
- QGPL/QPGMR: プログラミング・サブシステムで使用
- QGPL/QSPL: スプーリング・サブシステム印刷装置書出プログラムで使用
- QGPL/QSPL2: 基本システム・プールでの一般スプーリングで使用

実行時属性

以下は、実行管理機能に重要なクラス・オブジェクト内にある、実行時属性またはパラメーターの一部のリストです。

実行優先順位 (RUNPTY)
クラスを使用する実行中のすべてのジョブに割り当てられる優先順位レベルを指定する数。優先順位レベルは、システム・リソースのために競合しているすべてのジョブの中で、どのジョブを次に実行するかを決定するために使用されます。値は 1 から 99 の範囲で指定可能であり、1 が最高の優先順位です (1 の優先順位を持つすべてのジョブが最初に実行される)。この値は、ジョブ内のどのスレッドにも許可される最高の実行優先順位です。ジョブ内の個々のスレッドの優先順位は、さらに低い場合があります。ジョブの実行優先順位を変更すると、そのジョブ内のすべてのスレッドの実行優先順位が影響を受けます。 たとえば、ジョブが優先順位 10 で実行しており、ジョブ内のスレッド A は優先順位 10、ジョブ内のスレッド B は優先順位 15 で実行しているとします。ジョブの優先順位が 20 に変更されると、スレッド A の優先順位は 20 に調整され、スレッド B の優先順位は 25 に調整されます。

タイム・スライス (TIMESLICE)
ジョブ内の各スレッドに付与される最大のプロセッサー時間 (ミリ秒単位)。ジョブ内の他のスレッドまたは他のジョブに実行の機会が与えられる前にこのクラスが使用される。タイム・スライスは、意味を成す処理量を実現するために、ジョブ内のスレッドが必要とする時間を設定します。タイム・スライスの終了時に、他のスレッドが記憶域プールでアクティブ状態になるよう、スレッドが非アクティブ状態にされる場合があります。

デフォルトの待ち時間 (DFTWAIT)
待機実行の命令の完了をシステムが待つデフォルトの時間。この待ち時間は、命令がシステム・アクションを待機している時間に適用され、命令がユーザーからの応答を待機している時間には適用されません。通常、これは要求を終了する前にユーザーがシステムを待機する時間です。待ち時間を超過すると、エラー・メッセージがジョブに渡されます。このデフォルトの待ち時間は、特定の状況で待ち時間がこれ以外の方法で指定されていない場合に使用されます。

ファイル・リソースの割り振りに使用される待ち時間はファイル記述内に指定され、指定変更コマンドによって指定変更できます。これは、クラス・オブジェクトで指定されている待ち時間を使用することを指定します。ファイルを開くときにファイル・リソースが使用できない場合、システムはそれを待ち時間が終了するまで待ちます。

注: クラス属性は、ジョブの各経路指定ステップに適用されます。ほとんどのジョブは経路指定ステップを 1 つだけ持っていますが、ジョブが (「ジョブ再経路指定 (RRTJOB)」または「ジョブ転送 (TFRJOB)」コマンドなどにより) 再経路指定される場合は、クラス属性はリセットされます。

最大 CPU 時間 (CPUTIME)
ジョブの経路指定ステップが処理を完了するために許可される、最大のプロセッサー時間。ジョブの経路指定ステップがこの時間内に完了しない場合、それは終了し、メッセージがジョブ・ログに書き込まれます。

最大一時記憶 (MAXTMPSTG)
ジョブの経路指定ステップで使用できる最大の一時記憶域量。この一時記憶は、ジョブで実行するプログラム、ジョブをサポートするために使用されるシステム・オブジェクト、およびジョブによって作成される一時オブジェクトに使用されます。

最大スレッド数 (MAXTHD)
このクラス内のジョブがいつでも実行できるスレッドの最大数。複数のスレッドが同時に開始され

た場合、この値を超える可能性があります。超過したスレッドは、その通常の完了で実行できます。追加のスレッドの開始は、ジョブ内の最大スレッド数がこの最大値を下回るまで禁止されます。

注: スレッドが使用するリソースおよびシステム上の使用可能なリソースは変わる可能性があります。したがって、追加のスレッドの開始は、この最大値に達するまで禁止されることがあります。

関連タスク

145 ページの『クラス・オブジェクトの作成』
クラス・オブジェクトは、文字ベース・インターフェースを使用して作成できます。クラスは、クラスを使用するジョブの処理属性を定義します。ジョブが使用するクラスは、ジョブを開始するために使用されるサブシステム記述の経路指定項目で指定されます。ジョブが複数の経路指定ステップで構成されている場合、後続のそれぞれの経路指定ステップで使用されるクラスは、経路指定ステップを開始するために使用される経路指定項目の中で指定されます。

145 ページの『クラス・オブジェクトの変更』
クラス・オブジェクトの属性は、文字ベース・インターフェースを使用して変更できます。共通権限権限を除き、どの属性でも変更できます。オブジェクト権限の変更の詳しい情報については、オブジェクト権限取り消し (RVKOBJAUT) コマンドおよびオブジェクト権限付与 (GRTOBJAUT) コマンドを参照してください。

ジョブ・ユーザー ID:

ジョブ・ユーザー *ID (JUID)* は、あるジョブが他のジョブによって識別されるユーザー・プロファイルの名前です。この名前は、他のジョブがこのジョブを操作しようとする際の権限許可検査に使用されます。

他のジョブを操作する関数の例の一部として、サービス・ジョブ開始 (STRSRVJOB) コマンド、ジョブ情報の検索 (QUSRJOBI) API、ジョブの変更 (QWTCHGJB) API、すべてのジョブ制御コマンド、およびあるジョブから別のジョブにシグナルを送信する関数があります。

ジョブ間でユーザー・プロファイルをスワップする場合、現行ユーザー・プロファイルは、JUID ではなく、初期スレッドを実行中のプロファイルを識別します。

JUID は、ジョブ内で権限許可検査を行うのに使用されません。関数を実行する権限は、関数が呼び出されるスレッドの現行ユーザー・プロファイルに必ず基づきます。

ジョブがジョブ待ち行列または出力待ち行列にある場合、JUID はジョブのユーザー名と常時同じで、変更できません。

ジョブが開始され、後続の経路指定ステップが開始されると、JUID はジョブの現行ユーザー・プロファイルの名前と同じになります。ジョブがアクティブ状態にある間、JUID は以下の方法で変更できます。

- ジョブ・ユーザー ID の設定 (QWTSJUID) アプリケーション・プログラム・インターフェース (API) または QwtSetJuid() 関数を使用するアプリケーションによって、JUID を明示的に設定できます。JUID は、API または関数を呼び出したスレッドが実行されているユーザー・プロファイルの名前で設定されます。
- QWTSJUID API または QwtClearJuid() 関数を使用するアプリケーションによって、JUID を明示的に消去できます。その場合、ジョブは単一スレッドのジョブとして実行される必要があります。消去されると、システムによって JUID は、その時点でジョブの単一スレッドが実行されているユーザー・プロファイルの名前に暗黙的に設定されます。

- ジョブが単一スレッドのジョブとして実行されていて、アプリケーションによって明示的に JUID が設定されていない場合、ジョブがプロファイルの設定 (QWTSETP) API を使用して別のユーザー・プロファイルで実行されるたびに、システムは JUID を QWTSETP が設定したユーザー・プロファイルの名前に暗黙的に設定します。
- 単一スレッドのジョブが 2 次スレッドを開始し、アプリケーションが JUID を明示的に設定していない場合には、システムは JUID を、2 次スレッドが開始された時点でジョブの単一スレッドが実行されていたユーザー・プロファイルの名前に暗黙的に設定します。

  ジョブが単一スレッドに戻されると、システムは JUID を、ジョブの単一スレッドがその時点で実行されていたユーザー・プロファイルの名前に暗黙的に設定します。

  関連概念

  適切な権限
  ジョブの属性に変更を加えるには、ほとんどの場合、ユーザーがジョブ制御特殊権限 (*JOBCTL) を持っているか、ユーザー・プロファイルが、変更するジョブのジョブ・ユーザー ID と一致している必要があります。

ジョブ・ユーザー ID の例:

この例では、ジョブ・ユーザー ID (JUID) を様々な状態で割り当てる方法を示しています。

- ジョブが USERA という名前のユーザー・プロファイルで実行されます。JUID は USERA です。ジョブが QWTSETP API を使用して USERB に切り替えると、JUID は USERB に変更されます。

  この場合、JUID の「Set By (設定)」値は *DEFAULT です。ジョブは単一スレッドを実行中なので、ジョブ・ユーザー ID は、ジョブの初期スレッドを実行中の現行ユーザー・プロファイルです (ジョブ・ユーザー ID がアプリケーションによって明示的に設定された場合を除く)。ジョブ待ち行列ジョブおよび完了ジョブの場合、ジョブ・ユーザー ID は修飾ジョブ名のユーザー名です。
- 単一スレッドのジョブがユーザー・プロファイル USERX で実行します。JUID は USERX です。ジョブが 2 次スレッドを開始しても、JUID は引き続き USERX です。すべてのスレッドが USERY スワップされても、JUID はまだ USERX です。

  この場合、JUID の「Set By (設定)」値は *SYSTEM です。これは、現在マルチスレッドのジョブとして実行されているアクティブ状態にあるジョブですので、ジョブ・ユーザー ID はシステムによって暗黙的に設定されます。ジョブ・ユーザー ID は、ジョブがマルチスレッドになったときに実行されていたユーザー・プロファイルの名前に設定されます。ジョブが単一スレッドの実行に戻ると、ジョブ・ユーザー ID は *DEFAULT 値にリセットされます。
- SERVER というユーザー・プロファイルで実行されているサーバーが QWTSJUID API を呼び出すと、JUID は SERVER に設定されます。その後、サーバーがプロファイルの設定 (QWTSETP) API を呼び出し、代行処理作業中に現行ユーザー・プロファイルをクライアントのために CLIENT に設定する場合、JUID は引き続き SERVER です。同様に、それぞれが QWTSETP を呼び出して様々なユーザー・プロファイルで実行する 2 次スレッドをサーバーが開始する場合、JUID は SERVER のままです。

  この場合、JUID の「Set By (設定)」値は *APPLICATION です。ジョブ・ユーザー ID は、API を使用するアプリケーションによって明示的に設定されます。この値は、単一スレッドおよびマルチスレッドのジョブどちらにも適用されます。

スレッド:

スレッド という語は、「制御のスレッド」を省略したものです。スレッドとは、実行中にプログラムがたどるパス、実行されるステップ、およびステップが実行される順序のことです。スレッドは、特定の入力に対して事前定義した順序でコードを開始位置から実行します。

ジョブ内のスレッドを使用すると、一度にたくさんの処理を実行できます。たとえば、ジョブの処理中に、スレッドはジョブが処理を終了するのに必要なデータを検索し、計算することができます。

すべてのアクティブ・ジョブには、最低 1 つのスレッドがあり、これは初期スレッドと呼ばれます。初期スレッドは、ジョブの開始時に作成されます。System i ナビゲーター でスレッドを表示すると、デフォルトでは、 最初のスレッドのタイプとして、リストに**初期**と表示されます。 初期スレッドは、ジョブの開始時にジョブの中で作成される最初のスレッドです。

スレッド・タイプ

スレッド・タイプは、スレッドがシステム上で作成された方法を判別します。

ユーザー
: カスタマー・アプリケーションによって作成可能なスレッドです。ジョブの初期スレッドは、常にユーザー・スレッドです。複数のユーザー・スレッドを使用する場合は、「マルチスレッドの許可」フィールドを「はい」に設定することが必要です。

システム
: ユーザーの代わりにシステムによって作成されたスレッドです。一部のシステム機能は、システム・スレッドを使って処理を実行します。カスタマー・アプリケーションが、スレッドを使用するシステム機能を使う場合、システム・スレッドが使用されます。

関連タスク

147 ページの『スレッド・プロパティーの表示』
スレッドによって、ジョブは一度に複数の処理を行うことができます。スレッドが処理を停止すると、ジョブの実行も停止することがあります。

146 ページの『特定のジョブの下で実行しているスレッドの表示』
システムで実行中のすべてのアクティブ・ジョブでは、最低 1 つのスレッドが実行されています。スレッドは、ジョブと同じリソースを使用するジョブ内で実行する独立した作業単位です。ジョブはスレッドが実行する作業に依存するので、特定のジョブ内で実行するスレッドを検索する方法を理解することは重要です。

148 ページの『スレッドの終了または削除』
ジョブの開始時に作成される初期スレッドは、削除または終了することができません。しかし、ジョブが実行を続けられるように 2 次スレッドを終了しなければならないことがあります。終了しようとするスレッドには注意してください。その中で実行しているジョブはそのスレッドの機能がないと完了できないことがあるからです。

関連情報

例: スレッドを終了する (Java)

Thread management APIs

適切なスレッド権限:

スレッドを処理するには、特定の権限レベルが必要です。

スレッドの属性を表示および変更するには、ほとんどの場合、ユーザーが *JOBCTL 特殊権限を持っているか、ユーザー・プロファイルがスレッドを含むジョブのジョブ・ユーザー ID と一致している必要があ

ります。スレッドの実行優先順位を変更するには、*JOBCTL 特殊権限がなければなりません。スレッド制御権限があると、スレッドの属性の一部を表示できます。

スレッドを保留または開放するには、ユーザーが *JOBCTL 特殊権限またはスレッド制御権限を持っているか、ユーザー・プロファイルがスレッドを含むジョブのジョブ・ユーザー ID と一致している必要があります。スレッドを終了するには、*SERVICE 特殊権限、またはスレッド制御権限が必要です。

ライブラリー・リスト中のライブラリーなどの System i を参照するスレッド属性の場合、ユーザーには、そのオブジェクトに対する適切な権限が必要です。

i5/OS 権限について詳しくは、「Security Reference」の一連のトピックの『コマンドによって使用されるオブジェクトで必要な権限』を参照してください。

注: スレッド制御権限を使用して、他のジョブのスレッドに関する情報を検索することができます。 System i ナビゲーター のアプリケーション管理サポートまたは「機能使用法情報の変更」 (QSYCHFUI) API を、機能 ID を QIBM_SERVICE_THREAD に指定して使用することにより、スレッド制御を個々のユーザーに関して認可および取り消すことができます。 アプリケーション管理の詳細については、Information Center の『アプリケーション管理 (Application Administration)』というトピックを参照してください。

スレッド状況:

スレッドの現在の状況は、「詳細状況」の「スレッドのプロパティー」ウィンドウにある「汎用」ページに表示されます。

詳細状況には以下のような情報が示されます。

待ち行列解除の待機

ジョブのスレッドは、待ち行列解除操作の完了を待機しています。待ち行列解除は、待ち行列からメッセージを除去する操作です。メッセージは、1 人の人物または 1 つのプログラムから別の人またはプログラムに送信される通信内容です。特にメッセージは、1 つのスレッドによって待ち行列システム・オブジェクトにエンキュー (配置) され、別のスレッドによって待ち行列解除 (除去) されます。

注: 待ち行列解除の待機がプロパティー・ページに表示されている場合、待機中の待ち行列を識別する追加情報が表示されます。ジョブまたはスレッドが、i5/OS オブジェクトに対する待ち行列解除操作が完了するのを待機している場合、10 文字のオブジェクト名、そのライブラリー、およびオブジェクト・タイプが表示されます。ジョブまたはスレッドが内部オブジェクトに対する待ち行列解除操作が完了するのを待機している場合、30 文字のオブジェクト名が表示されます。内部オブジェクトの場合、30 文字の名前を表示するにはジョブ制御特殊権限 (*JOBCTL) が必要です。

詳細状況では、関連した状況値が表示され、スレッドの現在の状況に基づいた追加の情報が示されます。詳細状況に加えて表示される関連した状況値には、以下のようなものがあります。

保留 (n)

個々のスレッドが保留にされています。ジョブとは異なり、スレッドは一度に複数を保留にできます。スレッド状況に続く数値 (たとえば、(3)) は、スレッドが解放されずに保留された回数をユーザーに通知します。たとえば、スレッドで 3 回の保留が行われ、1 回解放があった場合は、まだ 2 回の保留があるということになります。数値は、状況が「プロパティー」ページに表示される場合にのみ表示され、リストに表示される場合は表示されません。スレッド処理を再開するには、スレッドの解放アクションを選択します。

さまざまなスレッド状況の詳細については、System i ナビゲーター のオンライン・ヘルプを参照してください。

ロック・オブジェクト:

ジョブおよびスレッドは、オブジェクトを使用して作業を処理します。

同時に複数の作業が処理されるので、オブジェクトにロックがかけられ、データの保全性が保たれます。*ロック・オブジェクト*は、ジョブおよびスレッドが作業を処理するために使用するシステム・オブジェクトです。ジョブおよびスレッドの実行が終了すると、オブジェクトはアンロックされ、さらに作業の処理が可能な状態になります。使用するロック要求タイプによっては、オブジェクトをロックすると、同時に 1 人のユーザーが 1 つのオブジェクトを使用することしか許可されません。たとえば、複数のユーザーが同時にあるオブジェクトを変更しようとすると、最初のユーザーがオブジェクトの更新を完了するまで、2 番目のユーザーがオブジェクトを変更できないようにロックします。ロック・ホルダーの使用によって、ユーザーはその時点で何がロックを保有しているか、または何がオブジェクトのロックを待っているかが分かります。

*有効範囲*は、ロックをジョブ、スレッド、ロック・スペースのうちのどれと関連付けるかを指定します。また有効範囲は、ロックが有効な期間と、オブジェクトが持つロック要求タイプおよび競合規則を定義します。

ロック要求タイプ は、ジョブ、スレッド、またはロック・スペースがロックされたオブジェクトに対して使用することのできる、異なるアクセス権のレベルです。たとえば、排他的ロック、読み取り不可ロック・タイプは、オブジェクトをシステム上で変更または削除している場合に使用されます。このロック要求タイプは、オブジェクトの使用または読み取りをだれにも許可しません。

種々のロック要求タイプは、以下のとおりです。

排他的 - 読み取り不可
オブジェクトは、排他的使用のために予約済みです。しかし、オブジェクトが任意のロック要求タイプによってロックされている場合、このオブジェクトを排他的に使用することはできません。このロック状態は、実行中の機能が完了するまで他のユーザーにこのオブジェクトにアクセスしてほしくない場合に適切です。

排他的 - 読み取り
オブジェクトは、共用読み取りロック要求タイプとのみ共用が可能です。このロックは、他のユーザーが読み取り以外の操作をしないようにする場合に適切です。

共用 - 更新
オブジェクトは、共用読み取りまたは共用更新ロック要求タイプのいずれかと共用できます。つまり、他のユーザーは同一のオブジェクトに対して、共用読み取りロック状態または共用更新ロック状態のいずれかを要求できます。このロック状態は、オブジェクトを変更するユーザーが、他のユーザーにもそのオブジェクトの読み取りまたは変更を許可する場合に適切です。

共用 - 更新不可
オブジェクトは、共用 - 更新不可、および共用 - 読み取りロック要求タイプとのみ共用できます。このロック状態は、オブジェクトを変更するつもりはなく、他のユーザーもこのオブジェクトを変更しないようにしたい場合に適切です。

共用 - 読み取り
オブジェクトは、排他的 - 読み取り不可を除くすべてのロック要求と共用できます。つまり、他のユーザーは、排他的 - 読み取り、共用 - 更新、共用 - 読み取り、または共用 - 更新不可ロック状態を要求できます。

*ロック状況*は、ロック要求の状態について通知します。種々のロック状況は、以下のとおりです。

保有 - ロック要求が実行され、ジョブ、スレッド、またはロック・スペースがロックを保有しています。

待機中 - ジョブまたはスレッドは、ロックの入手を待っています。

要求済み - ジョブまたはスレッドは、ロックを要求しました。

*ロック・ホルダー*は、現時点でロックを保有している、または特定のロック・オブジェクトでロックを待機しているジョブ、スレッド、およびロック・スペースです。

ジョブ・タイプ

システムはさまざまなジョブ・タイプを処理します。この情報では、それらのジョブについて、またその使い方が説明されています。

自動開始ジョブ:

自動開始ジョブは、繰り返し作業、特定のサブシステムと関連した一回限りの初期化作業を実行するバッチ・ジョブで、アプリケーション用の機能を初期化したり、同じサブシステム内の他のジョブ用の集中サービス機能を提供したりします。制御サブシステム内の自動開始ジョブは、(IBM 提供の制御サブシステムのように) 他のサブシステムを開始するために使用できます。サブシステムに関連付けられた自動開始ジョブは、サブシステムの各開始時に自動的に開始されます。

すべての自動開始ジョブはサブシステムの開始時に開始するので、サブシステムの最大ジョブ数に値を指定しておくと、自動開始ジョブが開始しなくなるという事態を防ぐことができます。サブシステムの最大ジョブ数を超えてしまうと、その他のジョブは開始できなくなります。相当数の自動開始ジョブが完了して、実行中のジョブの数が最大アクティビティー・レベルより少なくなると、サブシステムの他のジョブを開始できます。

自動開始ジョブに使用するジョブ記述は、「自動開始ジョブ項目追加 (ADDAJE)」コマンドを使用して指定します。 サブシステムの開始時に、ジョブは指定されたジョブ記述内のユーザー・プロファイル名で作動します。 USER(*RQD) を含むジョブ記述を指定することはできません。自動開始ジョブは、ジョブ記述で指定されるユーザー・プロファイルで作動するので、ジョブ記述を変更できる人物を制御する必要があります。

サブシステムに複数の自動開始ジョブが指定されている場合、自動開始ジョブは、順次ではなく、すべてが即時に開始します。サブシステムの最大ジョブ数を超えてしまうと、相当数の自動開始ジョブが完了して、実行中のジョブの数が最大アクティビティー・レベルを下回るまで、サブシステム内でその他のジョブは開始できなくなります。

バッチ・ジョブ:

バッチ・ジョブとは、事前定義された処理アクションのグループのことで、システムに投入され、ユーザーとシステムとの間の対話はほとんどあるいはまったくなしに実行されます。実行にユーザー対話を必要としないジョブは、バッチ・ジョブとして処理できます。一般にバッチ・ジョブは優先順位の低いジョブであり、実行するには特別なシステム環境が必要です。

バッチ・ジョブはシステムのバックグラウンドで実行され、ジョブを投入したユーザーが他の作業を行うように解放します。同時にいくつかのバッチ・ジョブをアクティブ状態にすることができます。

以下のリストでは、さまざまな種類のバッチ・ジョブを説明しています。

単純バッチ・ジョブ

単純なバッチ・ジョブは、ジョブ待ち行列に投入されるジョブです。これは、他のバッチ・ジョブと並んで待機し、その優先順位と順序番号に応じて処理されます。

バッチ即時ジョブ

バッチ即時ジョブは、その親ジョブの多くの属性とともに開始されるバッチ・ジョブです。このジョブは、親ジョブと同じサブシステムで実行されます。 (これは spawn() API の使用によって実現されます。)このジョブは親ジョブから属性をコピーして、ジョブ待ち行列には入れられないため、ジョブ待ち行列に入れられるジョブよりも速く開始できます。

バッチ MRT ジョブ

バッチ MRT ジョブとは、複数要求端末 (MRT) ジョブのことです。 MRT ジョブはサーバーのような機能を果たす S/36 環境ジョブで、他の S/36 環境ジョブが接続して MRT プロシージャーを実行できるようにします。

バッチ印刷ジョブ

バッチ印刷ジョブは、現行のユーザー・プロファイルが開始時のユーザー・プロファイルと異なるジョブによって作成されたプリンター出力ファイル (スプール・ファイルとも呼ばれる) をトラッキングします。

バッチ・ジョブは、ユーザーが以下を実行する場合に開始できます。

- ジョブをジョブ待ち行列に入れる
- 通信プログラム開始要求を発行する
- 事前開始ジョブでサブシステムを開始する
- spawn() API を使用する

バッチ・ジョブの開始方法:

ユーザーがバッチ・ジョブを投入すると、ジョブ待ち行列に入れられる前に、ジョブはいくつかのシステム・オブジェクトから情報を収集します。

1. ユーザーはジョブを投入します。
2. ジョブはジョブ属性を検索します。ジョブ投入 (SBMJOB) コマンドでジョブ属性が検出されない場合、ジョブは (SBMJOB コマンドで指定された) ジョブ記述、現行ユーザーのユーザー・プロファイル、および現行のアクティブ・ジョブ (SBMJOB コマンドを発行するジョブ) を参照します。

   注: 対話式ジョブ開始と同様、ユーザー・プロファイルを使用することをジョブ記述内に指定できます。ユーザー・プロファイルでは、特定のジョブ属性を検索するためにシステム値を使用することを指定できます。
3. ジョブがそのすべての属性を持つと、それはジョブ待ち行列上に置かれます。
4. サブシステムは、ジョブを処理できる状態になると、(そのサブシステムが割り振られている) ジョブ待ち行列内からジョブを探します。
5. 次いで、対話式ジョブ処理のように、サブシステムはそのジョブ記述から経路指定データについて検査します。
6. サブシステムは経路指定データを使用して、経路指定項目を検索します。経路指定項目は、ジョブが使用するプール、使用される経路指定プログラム、ジョブがその実行時属性を入手するクラスについての情報を提供します。
7. この情報の取得後に、経路指定プログラムが実行されます。 QCMD を使用する場合、QCMD は SBMJOB コマンドを実行します。 これは CMD または RQSDTA パラメーターで指定されたコマンドを実行します。

関連タスク

131 ページの『バッチ・ジョブの投入』
一般に、バッチ・ジョブは、実行に特殊なシステム環境を必要とする優先順位の低いジョブであるため (夜間に実行するなど)、バッチ・ジョブ待ち行列に入れられます。ジョブ待ち行列内で、バッチ・ジョブはランタイム・スケジュールおよび優先順位を受け取ります。ジョブをバッチ・ジョブ待ち行列に投入するには、文字ベース・インターフェースおよび以下の 2 つのコマンドのいずれかを使用します。

133 ページの『ジョブ待ち行列で待機中のバッチ・ジョブの開始』
ジョブが即時に開始するよう強制しなければならない場合があります。これを行う方法としては、使用中でないジョブ待ち行列にジョブを移動するのが最も効率的ですが、他の方法もあります。

関連情報

QPRTJOB ジョブ

***spawn* バッチ・ジョブ:**

spawn は、新しいジョブ・プロセス (子プロセス) を作成する機能です。作成されるプロセスは、呼び出しプロセス (親プロセス) の各種属性を継承します。新しいプログラムが指定されると、子プロセスで実行が開始します。バッチ・ジョブを spawn すると、親ジョブを使用して引数および環境変数が子ジョブに伝えられます。spawn() API は、バッチ即時ジョブ、事前開始ジョブ、または事前開始バッチ・ジョブを使用します。

関連情報

spawn()--spawn プロセス

SPAWN CL command, QUSRTOOL example

通信ジョブ:

通信ジョブは、リモート・システムからのプログラム開始要求によって開始されるバッチ・ジョブです。ジョブ処理には、通信要求および適切な指定が関係します。

通信バッチ・ジョブを i5/OS システム上で実行する場合、通信ジョブの作業項目が含まれるサブシステム記述がシステム上に存在している必要があります。 通信作業項目は、処理する通信ジョブのソースをサブシステムに示します。 ジョブ処理は、サブシステムが通信プログラム開始要求をリモート・システムから受け取り、その要求の適切な経路指定項目が検出されたときに開始されます。

通信ジョブの経路指定データ

通信ジョブのジョブ経路指定は、リモート・システムから受け取るプログラム開始要求によって決定されます。プログラム開始要求がターゲット・システム上で処理される場合、経路指定データとして使用される固定長データ・ストリームが作成されます。経路指定データの位置 25 には、通信要求用の PGMEVOKE が必ず含まれます。位置 29 の比較値 PGMEVOKE を指定するサブシステム経路指定項目には、プログラム名として一般に *RTGDTA があります。これは、(リモート・システムのプログラム開始要求からの) 経路指定データで指定されたプログラム名が、実行するプログラムであることを意味します。

特定の通信ジョブに特殊な処理環境が必要な場合、開始位置が 37 である比較値を指定して、サブシステム記述に付加的な経路指定項目を追加できます。この比較値には、プログラム開始要求のプログラム名を含める必要があります。この経路指定項目には、PGMEVOKE を比較値として使用する経路指定項目より低い順序番号が必要です。この方法により、特定の通信ジョブはさまざまなクラスまたはプール指定で実行できます。

セキュリティー

システムのセキュリティーでは、だれが通信装置を使用できるか、また関連した装置記述で使用されるコマンドをだれが使用できるかが制御されます。リモートおよびターゲット・システムの両方でアプリケーション・プログラムを作成および実行する場合は、追加のセキュリティー手段を考慮する必要があります。

通信ジョブのジョブ記述

通信ジョブに使用されるジョブ記述は、「通信項目追加 (ADDCMNE)」コマンドで指定されます。 このジョブ記述で指定されるユーザーは無視されます。システムは、プログラム開始要求から通信ジョブのユーザー名を入手します。プログラム開始要求がユーザー名を指定しない場合、システムは通信項目からのデフォルトのユーザー値を使用します。システム・セキュリティーの程度を引き上げるには、通信作業項目にデフォルトのユーザーを指定するのではなく、プログラム開始要求についてのユーザー情報を組み込んでください。

通信ジョブのタイプ:

このトピックでは、最も一般的なタイプの通信ジョブを説明しています。

Qlus (論理装置サービス)
Qlus は論理装置 (通信装置とも呼ばれる) のイベント処理を行います。 Qlus は正しい通信サブシステムへの装置の割り振りも行います。

Qcmnarbxx (通信アービター)
通信アービターは Qsysarb (システム・アービター) および Qtaparb (テープ・アービター) とともに、通信装置だけでなくすべての種類の装置の作業も処理します。この作業には、通信の接続、切断、装置ロック、およびエラー回復処理が含まれます。

再始動での通信アービター・ジョブ (QCMNARB) システム値は開始済みの通信アービター・ジョブの数を判別します。シングル・プロセッサー・システムでは最低で 3 つの通信アービターが開始されます。

Qsyscomm1 (システム通信)
このジョブはいくつかの通信および入出力 (I/O) アクティビティーを実行します。

Q400filsvr (リモート・ファイル・システム通信)
このジョブはリモート・ファイル・システムに対して共通プログラミング・インターフェース通信 (APPN または APPC) を実行します。

対話式ジョブ:

対話式ジョブとは、ユーザーがディスプレイ装置にサインオンした時点で開始され、ユーザーがサインオフした時点で終了するジョブのことです。このジョブを実行すると、ワークステーション項目またはユーザー・プロファイルで指定可能なジョブ記述をサブシステムが検索します。

対話式ジョブは、タスクを実行するために、ユーザーとシステムの間で両方向の連続した通信が必要です。対話式ジョブは、ユーザーがシステムにサインオンするときに開始されます。システムはサインオン情報を要求します。サインオン要求がシステムに受け入れられると、システムは対話式ジョブを作成します。 その後、システムはユーザーに要求を出すよう依頼します。ユーザーは要求を入力して、システムはその要求を処理することによって応答します。このパターンは、ユーザーがシステムをサインオフして対話式ジョブを終了するまで、またはアプリケーション例外または装置エラー・リカバリーのためにジョブが終了するまで繰り返します。

対話式ジョブがジョブ・グループまたはジョブの対の一部の場合、以下に示すジョブ・タイプのいずれかになります。

対話式 - グループ
対話式 - グループ・ジョブは、単一のディスプレイ装置に関連したジョブ・グループの一部です。

対話式 - システム要求
対話式 - システム要求ジョブは、システム要求機能によって互いに関連している 1 組のジョブの片方です。

ご存じでしたか? 2 つの方法でシステムにサインオンできます。ユーザー ID およびパスワードを使用して、システムに手動で入力できます。また、サーバーにユーザー ID およびパスワードを自動的に送信するプログラムを作成し、サインオン画面を省略することもできます。

対話式ジョブの開始方法:

ユーザーがシステムにサインオンすると、対話式ジョブが実行可能になる前に、サブシステムはいくつかのシステム・オブジェクトから情報を収集します。

1. サブシステムは、対話式ジョブの属性を入手するために、ワークステーション項目からジョブ記述を探します。ワークステーション項目がジョブ記述に *USRPRF を指定している場合、ジョブはユーザー・プロファイルからのその情報を使用します。

   注: この柔軟性により、ジョブの属性をワークステーションに関連付けるかまたは個々のユーザーに関連付けるかを指定できます。
2. サブシステムは、使用するジョブ記述を認識した後は、ジョブ記述内のすべてのジョブ属性は検索しない場合があります。一部の属性はユーザー・プロファイル内にある場合もあります。ユーザー・プロファイルに情報がない場合、サブシステムはシステム値を参照します。

   注: ユーザー・プロファイルにはジョブ属性が含まれており、特別にそのユーザーに合わせて特定の事項を調整できます。
3. サブシステムは、ジョブのすべての属性を収集した後に、新規対話式ジョブを開始できるか、またはエラー・メッセージをサインオン画面上で通知するかを決定します。 サブシステムは、サブシステムまたはワークステーション項目で許可されているジョブの最大数に達していないかどうかを検査します。次いで、有効なユーザー・プロファイル名が提供されていること、ユーザー・プロファイル名が使用可能なユーザー・プロファイルであること、および (必要であれば) 入力されたパスワードが有効であることを検査します。次いで、これはユーザーが、ジョブ記述、サブシステム記述、ワークステーション装置記述、および出力待ち行列とライブラリーに対する適切な権限を持っていることを検査します。最後に、サブシステムは、ユーザーがそのユーザー・プロファイルの許可されたサインオンの制限に達しているかどうかを検査します。 検証エラーが検出された場合、サインオン画面で適切なメッセージが表示されます。 検出されなかった場合は、対話式ジョブの開始プロセスが続行します。
4. サブシステムは、対話式ジョブが開始できることを検証した後に、ジョブ記述から経路指定データを検査します。サブシステムは経路指定データを使用して、サブシステム記述内から経路指定項目を検索します。経路指定項目は、ジョブが使用するプール、使用される経路指定プログラム、ジョブがその実行時属性を入手するクラスについての情報を提供します。
5. それらすべての情報が入手されると、経路指定プログラムが実行します。 IBM は QCMD と呼ばれる経路指定プログラムを提供していますが、これはすべてのタイプの作業に使用できます。 QCMD は、ジョブが対話式ジョブかどうかを把握し、ユーザー・プロファイルから実行する初期プログラムを確認します。初期プログラムが実行を完了すると、QCMD は初期メニューを表示します。

関連タスク

137 ページの『ワークステーションからの長時間実行機能の回避』
ワークステーションからの長時間実行機能 (保管/復元など) をタイアップなしで避けるために、システム・オペレーターはジョブをジョブ待ち行列に投入できます。

対話式ジョブの切断:

ジョブ切断 (DSCJOB) コマンドを呼び出すと、ジョブは切断され、サインオン画面が再表示されます。 ジョブに再接続するには、切断を実行した同じ装置にサインオンします。別の対話式ジョブが、異なるユーザー名の下の装置上で開始される場合があります。

- 「システム要求」メニューにあるオプションを使用すると対話式ジョブを切断でき、サインオン画面が表示されます。 オプションは DSCJOB コマンドを呼び出します。
- ジョブを再び接続すると、プログラム、メニュー、および現行ライブラリー用にサインオン画面で指定した値は無視されます。
- PC オーガナイザーまたは PC テキスト援助機能がアクティブになっているジョブを切断することはできません。
- TCP/IP TELNET ジョブは、セッションがユーザー指定の名前付き装置記述を使用している場合、切断可能です。以下のいずれか 1 つの方法を使用して、ユーザー指定の名前付き装置記述を作成できます。
  - DISPLAY NAME パラメーターを指定したネットワークステーションを使用する
  - System i Access PC 5250 Client Access サポートをワークステーション ID 機能と共に使用する
  - TCP/IP TELNET 装置初期化出口点を使用してワークステーション名を指定する
  - リモート装置パラメーターを指定して Telnet クライアント (STRTCPTELN) を使用する

  注: システム指定の装置名 (QPADEV* など) では、同じ装置にサインオンするのが同じユーザーである可能性はほとんどないため、ジョブを切断できません。
- グループ・ジョブに対してすべてのジョブを切断します。 再接続すると、切断を発行した場所に戻ります。最後のアクティブ・グループ・ジョブが再接続前に終了する場合、次のグループ・ジョブに戻ります。
- 何らかの理由でジョブを切断できない場合、そのジョブは終了します。
- サブシステム終了時には、サブシステムのうち切断されているすべてのジョブが終了します。サブシステムの終了中には、サブシステム内のどのジョブにおいても DSCJOB コマンドを発行することはできません。
- ジョブ切断間隔 (QDSCJOBITV) システム値は、ジョブを切断する時間間隔を指示するために使用できます。 この時間間隔に達すると、切断されたジョブは終了します。
- QDSCJOBITV 値を超えていない切断されたジョブは、サブシステムの終了時または IPL の実行時に終了します。

関連概念

137 ページの『ジョブ切断に関する考慮事項』
ジョブを切断する場合には必ず考慮する必要のある事柄がいくつかあります。

関連タスク

135 ページの『対話式ジョブの終了』
対話式ジョブを終了するには、いくつかの異なる方法を使うことができます。

136 ページの『装置からのすべてのジョブの切断』
ジョブ切断 (DSCJOB) コマンドにより、対話式ユーザーはワークステーションですべての対話式ジョブを切断して、サインオン画面に戻ることができます。 交換回線は、それがこのワークステーションのワークステーション装置記述で指定されており、その回線上の他のワークステーションはアクティブでな

い場合にのみドロップできます。切断ジョブのタイムアウト間隔 (QDSCJOBITV) システム値の切断間隔に達したときにジョブが切断されると、ジョブは終了し、ジョブ・ログはジョブのスプール出力には含まれません。

ジョブ要求元装置の入出力エラー:

要求元装置は、ユーザーがドメインにログオンし、ネットワーク・リソースを使用するためのワークステーションです。装置回復アクション (DEVRCYACN) ジョブ属性は、入出力エラーが発生した場合にジョブの要求元装置に対して実行するアクションを指定します。

DEVRCYACN 属性には以下のオプションがあります。

***SYSVAL**
これはデフォルトです。これは、装置エラーが起きた場合に実行するアクションを、ワークステーション (QDEVRCYACN) システム値に指定します。システム値は、ジョブ属性がサポートするすべての値をサポートします (*SYSVAL を除く)。

***MSG** 入出力エラー・メッセージを通知し、アプリケーション・プログラムにエラー回復を実行させます。この設定はお勧めしません。

***DSCMSG**
ジョブを切断します。これは出荷時のデフォルトです。再接続時に、新規エラー・メッセージがユーザーのアプリケーション・プログラムに通知を出し、入出力、および画面の内容を再表示する必要があるために、装置がエラーとなったが回復されたことを示します。

***DSCENDRQS**
ジョブを切断します。再接続時に、要求終了機能が実行され、ジョブの制御を最終要求レベルに戻します。

***ENDJOB**
ジョブを終了します。ジョブに対してジョブ・ログが生成される場合があります。メッセージがジョブ・ログと QHST ログに送信され、装置エラーが原因でジョブが終了したことを示します。

***ENDJOBNOLIST**
ジョブを終了します。ジョブ・ログは生成されません。メッセージが QHST ログに送信され、装置エラーが原因でジョブが終了したことを示します。

注: *DSCENDRQS、*ENDJOB、または *ENDJOBNOLIST が DEVRCYACN に対して指定されている場合、装置上でエラーが起きたときに回復アクションが有効になります。 他の値の 1 つが指定されている場合、エラーが生じた装置への次回の入出力時に回復アクションが実行されます。

対話式ジョブおよび経路指定ステップ:

初期メニューが呼び出される前に、経路指定データはサブシステム記述の中の経路指定項目と比較されます。一致すると、経路指定項目で指定されたプログラムが呼び出され、経路指定ステップが開始されます。

以下に、経路指定ステップが開始され、初期プログラムを指定するユーザー・プロファイルの初期メニューが表示されるまでの一連のアクティビティーを示します。

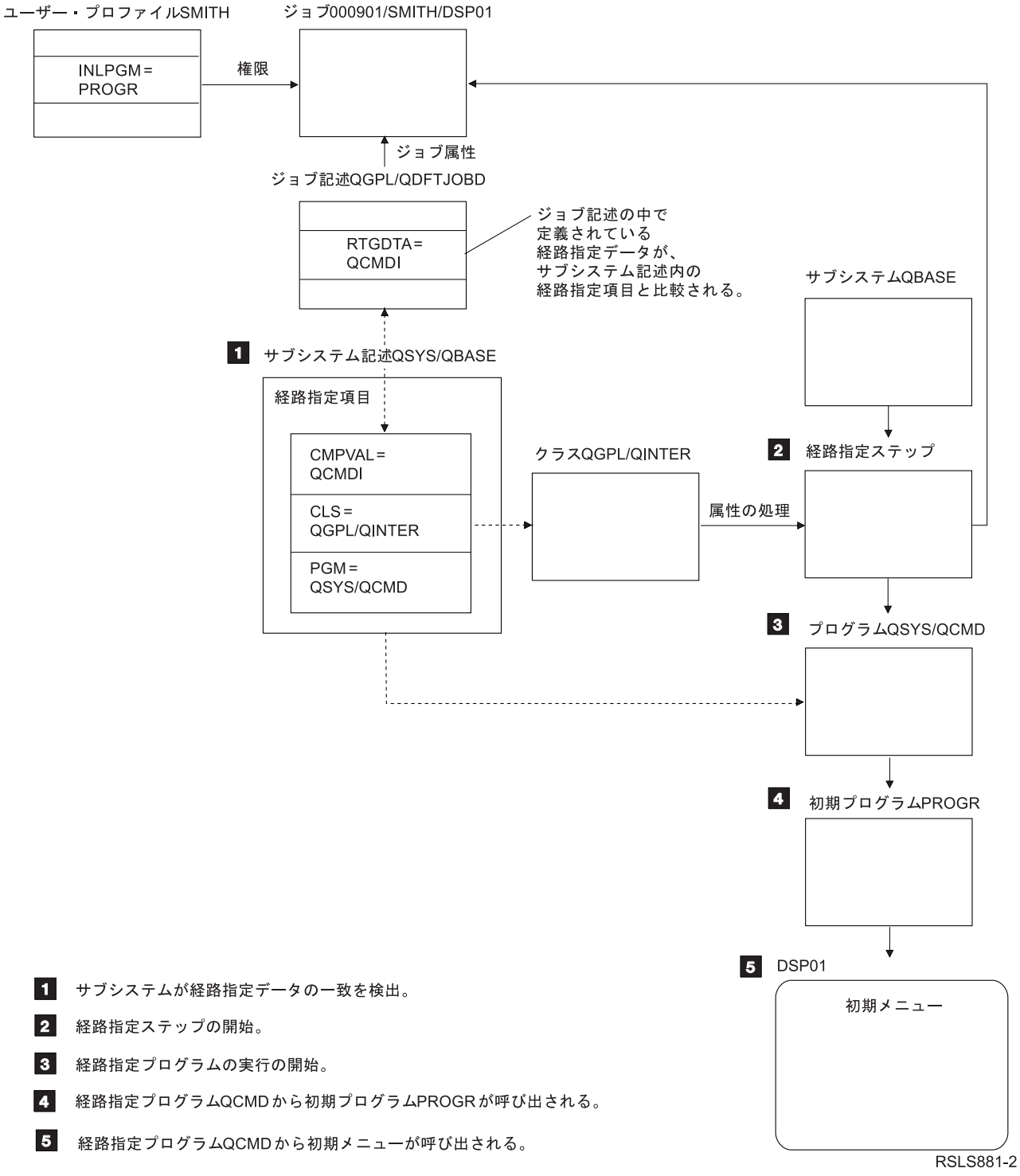


図 1. サブシステム・アクティビティー

対話式ジョブのアプローチ

対話式ジョブを多様な方法で扱うことができます。こうしたアプローチは、経路指定ステップを制御する方法に依存します。最初に、以下について判別してください。

- 経路指定ステップを制御するのは、QSYS/QCMD またはユーザー・プログラムのどちらのプログラムですか?

- 経路指定は、ユーザー・ベースそれともワークステーション・ベースでしょうか?

経路指定ステップを制御するプログラム:

特定のジョブに対する最善の方法を判別するには、経路指定ステップを制御するプログラムを最初に決定する必要があります。

対話式ジョブに対する QSYS/CMD の使用 - 利点

IBM 提供のコマンド・プロセッサー QSYS/QCMD は、ワークステーション・ユーザーが様々な機能を使用できるようにするという点ですばらしい柔軟性を提供しています。経路指定ステップを制御するために QCMD を使用することには、以下の利点があります。

- ユーザー・プロファイルで指定されている場合には、アテンション・プログラムがアクティブになります。
- ユーザー・プロファイルで指定されている初期プログラムが呼び出されます。
- ユーザー・プロファイルで指定されている初期メニューが呼び出されます。
- ユーザー・プロファイルで指定されているとおりに、System/36 環境にユーザーが配置されます。

さらに、デフォルトの QCMD を使用すると、ユーザー作成機能を呼び出すのに使用する CALL コマンドを含むコマンドを直接入力できるメインメニューに移動します。システム機能に簡単にアクセスできるようオンライン・ヘルプのあるメニュー・オプションが備えられています。さらに、コマンド選択メニュー、見出し検索への素早いアクセス、および (CALL QCMD と呼ばれる) コマンド入力機能も提供されています。コマンド入力機能は、コマンドを直接使用して多数の機能を用いる必要のあるプログラマーおよびオペレーターを主な対象としています。

対話式ジョブに対するユーザー・プログラムの直接的呼び出し - 利点

対話式ジョブの経路指定ステップを制御するために、ご使用のプログラムを直接呼び出すことができます。こうしたプログラムは、IBM 提供のプログラムに比べ、ワークステーション・ユーザーが必要とする機能へのより特化したアクセスを提供するように設計されています。加えて、そうしたプログラムは特定の機能用に編成されているので、必要なシステム・リソースが IBM 提供のプログラムよりも一般に少なくてすみます。また、初期プログラムや初期メニューなどの機能を提供したい場合もあります。

ワークステーション・ベースとユーザー・ベースの経路指定の比較:

経路指定ステップを制御するプログラムを判別した後、経路指定が、ジョブを開始したワークステーションに基づくのか、サインオンしたユーザー (ユーザー・プロファイル) に基づくのかを判別する必要があります。

ワークステーションに基づく経路指定は、装置のワークステーション項目またはプロファイルに関連付けられた、ジョブ記述内で指定された経路指定データを使用して実行されます。ユーザーに基づく経路指定は、ユーザー・プロファイル内またはユーザー・プロファイルのジョブ記述で指定された初期プログラムを使用して行うことができ、QCMD 以外の経路指定項目にマッピングされます。

初期プログラムの使用

初期プログラムは、ワークステーションと対話してワークステーション・ユーザーから入力値を取得できます。初期プログラムが呼び出される場合には、パラメーター値を受け取ることはできません。初期プログラムは、以下の 2 つの方法のいずれかで使用できます。

- ユーザー入力コマンド用の初期環境を確立する。たとえば、ライブラリー・リストが変更されたり、印刷ファイルおよびメッセージ・ファイルが指定変更される可能性があります。初期プログラムが機能を完了して QSYS/QCMD に戻ると、初期メニューが表示されます。
- ジョブの制御プログラムとして。初期プログラムが QSYS/QCMD に戻らないと、経路指定ステップの制御プログラムとなります。初期メニューは表示されません。ユーザーは、初期プログラムを介して使用可能な機能のみを要求できます。

  たとえば、特定のアプリケーション・オプションのあるメニューを表示できます。ユーザーは、メニュー上の機能のみを実行できます。こうしたオプションの一例には、サインオフがあります。SIGNOFF コマンドを実行すると、ジョブは終了し、システムのメインメニューは決して表示されません。このアプローチを使用する場合、ユーザー・プロファイル・オプション INLMNU を使用してメニューが表示されないようにすることを考慮してください。

初期プログラムを作成し、戻りが送信される場合に QSYS/QCMD に戻るようにもまたは戻らないようにもできます。初期プログラムが QSYS/QCMD に戻ると、初期メニューが表示されます。

同時に複数のジョブが終了する場合:

複数のジョブが同時に終了することが時折あります。たとえば、ネットワーク・エラーが生じて、ジョブ属性が *ENDJOB または *ENDJOBNOLIST に設定されます。ジョブの終了に加えて、以下の装置リカバリー・アクションが生じます。

- ジョブの優先順位が下がります。これにより、このジョブは他のアクティブ状態にある対話式ジョブと同じ優先順位を持たなくなります。
- ジョブのタイム・スライスは、100 ミリ秒に設定されます。これにより、高位の優先順位を持つジョブに、処理リソースを取得する十分な可能性が与えられることになります。

ジョブ属性が *ENDJOB または *ENDJOBNOLIST に設定されているジョブのジョブ・ログは、ジョブ・ログ保留状態になります。ジョブ・ログ保留状態にあるジョブ・ログからプリンター出力を生成するには、ジョブ・ログ表示 (DSPJOBLOG) コマンドを使用します。

ジョブの終了時にスプール・ファイルにジョブ・ログが書き込まれる方法を制御できます。ジョブ終了時に、ジョブ自体が書き込むか、バックグラウンド・サーバー・ジョブが行うか、全く書き込まないかのいずれかです。指定した値は、同時に多くのジョブが終了する際、全体の回復時間に影響を大きく及ぼすことがあります。 詳しくは、関連概念のジョブ・ログ保留についてを参照してください。

関連概念

87 ページの『ジョブ・ログ保留』
ジョブ・ログ保留状態は、これまで何年も使用されてきました。 ジョブのジョブ・ログ属性が *PND の場合、ジョブ・ログは生成されません。特定のジョブのジョブ・ログをどのように、またどのような状況下で生成するかを制御することができます。

事前開始ジョブ:

事前開始ジョブは、作業要求を受け取る前に実行を開始するバッチ・ジョブです。事前開始ジョブは、サブシステム内の他のタイプのジョブに先立って開始されます。事前開始ジョブは、事前開始ジョブ項目 (サブシステム記述の一部) を使用して、開始時に用いるプログラム、クラス、および記憶域プールを判別するので他のジョブとは異なります。

事前開始ジョブ項目で、事前開始ジョブのプールを作成および管理するためにサブシステムが使用する属性を指定する必要があります。作業要求を処理するのに必要な時間を減らすために、事前開始ジョブを使用します。事前開始ジョブには二つの種類があります。それぞれのタイプが処理する要求は異なります。

「Prestart (事前開始)」とだけ表示され、その後ジョブがそれ自身の最初の要求を待機しますが、それはその時点ではどの種類の要求をジョブが処理するのかをシステムが認識していないためです。

事前開始通信
このジョブは、遠隔システムがプログラム開始要求を送信する前に実行を開始する通信バッチ・ジョブです。

事前開始バッチ
このジョブは、作業要求を受け取る前に開始するバッチ・ジョブです。

事前開始ジョブは作業要求を受け取る前、つまりサブシステムが開始したときまたは「事前開始ジョブ開始」(STRPJ) コマンドの結果として開始します。事前開始ジョブはサブシステム記述の事前開始ジョブ項目 (PJE) から開始します。事前開始ジョブ項目は、事前開始ジョブでどのプログラムを実行するか、事前開始ジョブが実行を開始するときに使うユーザー・プロファイル、ジョブ記述、ジョブの実行時属性を指定するときに使うクラス、および事前開始ジョブを実行するメモリー・プールなどの属性を指定します。

事前開始ジョブは、作業要求を受け取る前に、開始して事前開始ジョブ自身を初期化します。これにより、要求を処理するのに必要な時間が少なくて済みます。事前開始ジョブを使用すると、一度初期化しただけで多数の要求を処理できるようになるため、要求ごとに新しいジョブが必要とされることはありません。多くのクライアント・サーバー・アプリケーションが、クライアント・ユーザーの要求を処理するために事前開始ジョブを使用しています。このようにジョブをすぐに実行できるようにしておくなら、事前開始ジョブがユーザーの要求の処理を即時に開始できるので、パフォーマンスが向上します。

注: サブシステムの最大ジョブ数に値を指定しておくと、事前開始ジョブが開始されないようにすることができます。サブシステムの最大ジョブ数を超えた場合、事前開始ジョブは開始できなくなります。相当数のジョブが完了して、実行中のジョブの数がサブシステムの最大ジョブ数以下になると、サブシステムの事前開始ジョブを開始することができます。

プログラム開始要求

プログラム開始要求 (PSR) は、SNA サーバーに接続するための SNA クライアント用に設計されています。事前開始ジョブが PSR を処理するためにセットアップされると、ジョブの外部状態は PSRW (プログラム開始要求待機中) です。

さらに、事前開始ジョブは最も有名なホスト・サーバーである IBM 提供の TCP/IP サーバーにも使用します。こうした事前開始ジョブは、内部インターフェースを介して作業を受け入れ、PSR は使用されません。しかし、作業を待機中の事前開始ジョブは、PSR を使用していない場合であっても、引き続き PSRW 状態が表示されます。

関連概念

101 ページの『事前開始通信ジョブおよびジョブ会計』
システムでジョブ会計を使用する場合、プログラム開始要求が事前開始ジョブに添付された直後に、事前開始ジョブの変更 (CHGPJ) コマンドを会計コード・パラメーター用のプログラム開始要求値 (CHGPJ ACGCDE(*PGMSTRRQS)) を使用して実行する必要があります。

関連タスク

138 ページの『事前開始ジョブの開始』
一般的な事前開始ジョブは、サブシステムの開始と同時に開始されます。エラーのためすべての事前開始ジョブがシステムによって終了された場合、または事前開始ジョブ項目の STRJOBS (*NO) のためにサブシステムの開始時にすべての事前開始ジョブが開始されなかった場合には、事前開始ジョブを手動で開始します。事前開始ジョブを開始するには、文字ベース・インターフェースを使用します。

144 ページの『事前開始ジョブの終了』
アクティブ・サブシステムの事前開始ジョブを終了するには、文字ベース・インターフェースを使用できます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

事前開始ジョブ名:

事前開始ジョブの完全修飾された 3 部構成の名前は、一度事前開始ジョブが開始されると絶対に変更できません。完全修飾された 3 部構成のジョブ名のユーザー名には、事前開始ジョブが開始されるユーザー・プロファイルが必ず含まれます。

事前開始ジョブが作業要求を処理する前にスプール・ファイルがオープンされると、そのスプール・ファイルは事前開始ジョブ項目のユーザー・プロファイルに関連付けられます。 その他の場合には、ジョブの現行のユーザー・プロファイルに関連付けられます。

事前開始ジョブ項目のプロファイルと現行ユーザー・プロファイルが異なる場合には、スプール・ファイルは QPRTJOB というジョブ名で、現行ユーザー・プロファイルのユーザー名を持つジョブでスプールされます。(これは、サーバー・ジョブの事前開始ジョブ項目にも当てはまります。)

事前開始ジョブ項目のクラス (CLS) パラメーターによって、1 つの事前開始ジョブ項目ごとに 2 つある事前開始ジョブ・クラスのパフォーマンス特性を制御する方法が提供されます。

事前開始ジョブの機能の仕方:

事前開始ジョブは、作業の着手前に開始されるジョブです。これによりシステムは、新しいジョブの開始を原因とする遅延なしで、作業に対する要求を処理できます。

事前開始ジョブは、ユニークなタイプのバッチ・ジョブです。これは、このジョブがジョブ・タイプ 'B' を持ち、ジョブ・サブタイプ 'J' を持つという意味です。この拡張ジョブ・タイプはさらに、ジョブを事前開始ジョブ (1610)、事前開始バッチ・ジョブ (1620)、または事前開始通信ジョブ (1630) として定義します。拡張ジョブ・タイプは、事前開始ジョブが作業要求を受け入れる方法を説明します。事前開始ジョブで実行するプログラムが、作業の受け入れに通信インターフェースを使用する場合、そのジョブは事前開始通信ジョブです。事前開始ジョブで実行するプログラムが、バッチ作業インターフェースから作業を受け入れる場合、そのジョブは事前開始バッチ・ジョブです。プログラムが作業の受け入れ段階まで達していない場合、そのジョブは単なる事前開始ジョブです。事前開始バッチ・ジョブは作業要求にサービスを提供するので、多くの場合サーバー・ジョブと呼ばれます。

通信作業要求は、割り振られた必須の通信装置を持つサブシステムにより処理されます。バッチ作業要求は一般に、システム QSYSWRK、QUSRWRK、または QSERVER に同梱される基本サブシステムの 1 つにより処理されます。

事前開始ジョブは、事前開始ジョブ項目に含まれる情報に基づいて開始されます。「事前開始ジョブ項目追加 (ADDPJE)」コマンドと「事前開始ジョブ項目変更 (CHGPJE)」コマンドの「ジョブ開始 (STRJOBS)」パラメーターは、事前開始ジョブをサブシステムの開始時に開始するか、または「事前開始ジョブ開始 (STRPJ)」コマンドの入力時に開始するかを指定できます。「ジョブ初期数 (INLJOBS)」パラメーターは、プログラムに先だって開始される事前開始ジョブの数を決定します。

作業要求の着信に応じて、さらに多くの事前開始ジョブが必要になる場合があります。「事前開始ジョブ項目追加 (ADDPJE)」コマンドと「事前開始ジョブ項目変更 (CHGPJE)」コマンドの「しきい値 (THRESHOLD)」パラメーターは、さらに多くのジョブを開始する時点を指示します。要求を処理するため

に使用できる事前開始ジョブの数が、THRESHOLD パラメーターで指定される値を下回ると、追加のジョブが開始されます。 「追加ジョブ数 (ADLJOBS)」パラメーターは、さらにいくつのジョブを開始できるかを指示します。

事前開始ジョブによっては、ある作業要求を処理してから、別の作業要求を処理するために使用できるものがあります。「最大使用数 (MAXUSE)」パラメーターにより、これらの事前開始ジョブが処理する作業要求の数を指定できます。 事前開始ジョブによっては、単一の作業要求を処理した後に終了し、MAXUSE 値は無視するものもあります。 事前開始ジョブが複数の作業要求を処理するか、または単一の作業要求のみを処理するかは、事前開始ジョブで実行するプログラムによって決定されます。

事前開始ジョブが少なくとも 1 つの作業要求を処理した後に終了する場合、サブシステムは、実行を継続しているジョブの数と INLJOBS パラメーターに指定されている数とを比較します。 実行を継続しているジョブの数が INLJOBS を下回る場合、サブシステムは別のジョブを開始します。

事前開始ジョブが少なくとも 1 つの作業要求をも処理することなく終了し、ジョブが「ジョブ終了 (ENDJOB)」コマンドで終了していない場合、事前開始ジョブ・プログラムはエラー状態であると見なされます。 サブシステムは事前開始ジョブ項目を、制御された方法で終了します。これにより、作業要求を扱うジョブはその要求を完了して、サブシステムによる追加のジョブの開始を回避できます。

サブシステムは使用可能な事前開始ジョブが多すぎないかを判別するために、事前開始ジョブの数を定期的に検査します。事前開始ジョブは、作業要求の待機時に使用できます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

経験報告: サブシステム構成

事前開始ジョブ項目:

事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

プログラム開始要求が事前開始ジョブに添付されている場合、事前開始ジョブのジョブ属性はサブシステムによって変更されません。しかし通常は、サーバー・ジョブはジョブ属性を、スワップされたユーザー・プロファイルのジョブ属性に変更します。

事前開始ジョブの変更 (CHGPJ) コマンドを使用すると、事前開始ジョブはジョブ属性の一部を (プログラム開始要求のユーザー・プロファイルに関連付けられたジョブ記述で指定された、または事前開始ジョブ項目で定められたジョブ記述で指定の) ジョブ記述のジョブ属性に変更できます。

関連概念

246 ページの『事前開始ジョブの調査』
このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

関連タスク

184 ページの『事前開始ジョブ項目の追加』
事前開始ジョブ項目は、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの入力時に開始できる事前開始ジョブを示します。 事前開始ジョブ項目は、文字ベース・インターフェースを使用してサブシステム記述に追加できます。

189 ページの『事前開始ジョブ項目の変更』
指定されたサブシステム記述内の事前開始ジョブ項目は変更可能です。事前開始ジョブ項目の変更時に、サブシステムはアクティブであってもかまいません。サブシステムがアクティブであるときに項目

に加えられた変更は、時間を経て反映されます。コマンドの発行後に開始されたすべての新規事前開始ジョブは、新規のジョブ関連値を使用します。このコマンドは、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの発行時に開始できる事前開始ジョブを識別します。

193 ページの『事前開始ジョブ項目の除去』
文字ベース・インターフェースを使用して、事前開始ジョブ項目をサブシステム記述から除去できます。現在アクティブ状態のいずれかのジョブが事前開始ジョブ項目を使用して開始された場合には、その項目を除去することはできません。

関連情報

Experience Report: 事前開始ジョブ項目の調整

事前開始ジョブ処理プログラムの開始要求:

事前開始ジョブが開始される場合、事前開始ジョブのユーザー・プロファイルで実行されます。プログラム開始要求が事前開始ジョブに添付されている場合には、事前開始ジョブのユーザー・プロファイルはプログラム開始要求のユーザー・プロファイルに置き換えられます。事前開始ジョブがプログラム開始要求の処理を終えると、プログラム開始要求のユーザー・プロファイルは事前開始ジョブのユーザー・プロファイルに置き換えられます。ユーザー・プロファイルに関連付けられたグループ・プロファイルがある場合、グループ・プロファイルもスワップされます。

スワップされたユーザー・プロファイルの唯一の目的は、権限検査です。ユーザー・プロファイルに関連する他の属性はスワップされません。事前開始ジョブ項目のユーザー・プロファイルが権限が与えられているライブラリー・リスト上のライブラリーは、プログラム開始要求のユーザー・プロファイルが事前開始ジョブ項目のユーザー・プロファイルに置換される際にその事前開始ジョブに引き続き権限を与えます。ライブラリー・リストは、ライブラリー・リスト変更 (CHGLIBL) コマンドで変更できます。

プログラム開始要求のための事前開始ジョブのオブジェクト権限

事前開始ジョブが開始されると、ジョブを開始する必要のあるすべてのオブジェクトで、事前開始ジョブ項目のユーザー・プロファイルに対して権限検査が実行されます。プログラム開始要求が事前開始ジョブに添付できるようになる前に、プログラム開始要求のユーザー・プロファイル/パスワードおよび通信装置に対する権限、さらにはライブラリー/プログラムのみが検査されます。

事前開始ジョブ項目のユーザー・プロファイルで権限が与えれているものの、プログラム開始要求のユーザー・プロファイルでは権限が与えられていないオブジェクトが存在しないようにするには、少なくとも事前開始ジョブ項目のユーザー・プロファイルで権限が与えられているオブジェクトにはプログラム開始要求のユーザー・プロファイルでも権限を与えるようにしなければなりません。このためには、CRTxxxPGM (xxx はプログラム言語) コマンドで USRPRF (*OWNER) が指定された事前開始ジョブ項目ユーザーによって、事前開始ジョブ・プログラムを作成できます。プログラム所有者権限は、事前開始ジョブ・プログラムによって呼び出される任意のプログラムに自動的に転送されます。別の方法としては、任意のオブジェクトを参照する前に、明示的にオブジェクト検査権限 (CHKOBJ) を選択できます。

事前開始ジョブのユーザー・プロファイルが権限を与えられていないファイルおよびオブジェクトは、リクエスター装置でトランザクションの終了が実行される前に、クローズされて割り振り解除されます。データベース・ファイルが事前開始ジョブでオープンされたままの場合、データベース・セキュリティーを保証するため、事前開始ジョブ・プログラムはプログラム開始要求のファイルをオープンするユーザー・プロファイル権限を検査する必要があります。

バッチ・アプリケーションの事前開始ジョブ:

事前開始ジョブおよびそれを使用するサーバー・ジョブは、ジョブ会計に関する固有の状態を提示します。1 つの事前開始ジョブがさまざまなユーザーを保守する場合は、使用されるリソースに関してそれらの各ユーザーに課金することもできます。それを行う場合は、毎回サービス要求を行う前と後に、会計コードを更新する必要があります。

ジョブ会計と事前開始ジョブの関連について詳しくは、101 ページの『事前開始通信ジョブおよびジョブ会計』を参照してください。

事前開始ジョブのパフォーマンスについてのヒント:

事前開始ジョブは、ICF プログラム装置の獲得や CPI 通信会話の受信を行う前に、可能な限り多くの作業を行います。できるだけ多くの作業 (オブジェクトの割り振り、データベース・ファイルのオープンなど) を最初に行うと、プログラム開始要求を受け取る際に行う必要のある作業が少なくなり、トランザクションの応答時間を短くできます。 以下に、事前開始ジョブを使用する際のパフォーマンスに関する別の考慮事項を幾つか取り上げます。

要確認: サブシステム内にアクティブ状態の事前開始ジョブ項目がある場合、定期的にサブシステムはプログラム開始要求を保守可能な状態にある、プール内の事前開始ジョブの数を検査して、使用可能な事前開始ジョブが過剰にあるかどうかを判別します。サブシステムは、過剰になっている使用可能な事前開始ジョブを徐々に終了します。しかし、サブシステムは少なくともプール内の INLJOBS 属性で指定されている事前開始ジョブの数は常に保持します。

- 実行するトランザクションに固有のリソースのみを割り振り解除してください。事前開始ジョブ・プログラムが実行する他のトランザクションで共用しているリソースは、ジョブが次の要求を待機している間は割り振られた状態のままにする必要があります。次の要求を受け取る際に時間を節約するためには、ファイルをオープンしたままに、オブジェクトも割り振られた状態のままにしておきます。

  注: 事前開始ジョブでオープンされたままになっているデータベース・ファイルについては、同じジョブで共用されているデータベース・ファイルと同じ考慮事項が必要になります。
- 事前開始ジョブの存在中すべてで同じ QTEMP ライブラリーが使用されるので、必要のなくなったオブジェクトは削除してください。
- 事前開始ジョブの存在中すべてで同じローカル・データ域 (LDA) が使用されるので、情報を保持して、次のトランザクションに渡してください。
- 各事前開始ジョブは多くのプログラム開始要求を処理できる一方、有するジョブ・ログは 1 つずつだけなので、アプリケーションが事前開始ジョブのアクティビティーを示すジョブ・ログにメッセージを送信するようにしたい場合もあります。また、バッチ事前開始ジョブのジョブ・ログは使用と使用の間に消去されるのでこれは有用です。
- プログラム開始要求が事前開始ジョブに添付されている場合、事前開始ジョブのジョブ属性はサブシステムによって変更されません。事前開始ジョブの変更 (CHGPJ) コマンドを使用すると、事前開始ジョブはジョブ属性の一部を (プログラム開始要求のユーザー・プロファイルに関連付けられたジョブ記述で指定された、または事前開始ジョブ項目で指定されたジョブ記述で指定された) ジョブ記述のジョブ属性に変更できます。
- 事前開始ジョブ項目のクラス (CLS) パラメーターによって、1 つの事前開始ジョブ項目ごとに 2 つある事前開始ジョブ・クラスのパフォーマンス特性を制御する方法が提供されます。たとえば、システムの使用率が既に高い場合に到着する作業には、低い実行優先順位を提供できます。

スプール・ファイルと事前開始ジョブ項目:

事前開始ジョブがプログラム開始要求を処理する前にスプール・ファイルがオープンされると、そのスプール・ファイルは事前開始ジョブ項目のユーザー・プロファイルに関連付けられます。オープンされない場合には、現行のプログラム開始要求のユーザー・プロファイルに関連付けられます。

事前開始ジョブ項目のプロファイルと現行プログラム開始要求のユーザー・プロファイルが異なる場合には、スプール・ファイルは QPRTJOB という 3 部構成のジョブ名の最初の部分と、ユーザー・プロファイルの名前となる 2 番目の部分を有するジョブでスプールされます。

読み取りプログラムおよび書き出しプログラムのジョブ:

*読み取りプログラム・ジョブ*はスプールされた入力ジョブで、書き出しプログラム・ジョブはスプールされた出力ジョブです。

読み取りプログラム
読み取りプログラム・ジョブは、データベース・ファイルからバッチ・ジョブ・ストリームを読み取り、ジョブ待ち行列にジョブを配置します。読み取りプログラム・ジョブは入力スプーリングの一部であり、 IBM 提供のプログラムです。

書き出しプログラム
書き出しプログラム・ジョブは、プリンター出力ファイル (スプール・ファイルとも呼ばれる) のレコードをプリンターに書き出します。書き出しプログラム・ジョブは IBM 提供のプログラムであり、スプーリング・サブシステムで開始され、そのサブシステムで、出力待ち行列から印刷するファイルが選択されます。

サーバー・ジョブ:

サーバー・ジョブは、システム上のバックグラウンドで継続的に実行するジョブです。

作業は、ユーザーの代わりをするネットワーク機能、オペレーティング・システム機能、ネットワーク内の別のシステム、クラスター・サーバー・ジョブなどの汎用システム・サービスから入ってきます。サーバー・ジョブは通常、システム出荷時の 3 つの基本サブシステムにある、 QSYSWRK、 QSERVER、 QUSRWRK のいずれかで処理されます。一般にサーバー・ジョブは、HTTP、Lotus Notes、TCP/IP などの機能と関連しています。システムには、サーバー・ジョブ用に 3 つの基本モデルがあります。

スレッド化ジョブ・モデル
スレッド化ジョブ・モデルでは、サーバー・ジョブは複数のスレッドを持つジョブになります。 1 つのスレッドが、他のスレッドに対して作業の分配役を果たします。たとえば、サーバーがクライアント要求を受け取る場合、初期スレッドは要求を読み取り、別のスレッドにそれを渡して要求を処理します。このモデルを使用すると、システム上のジョブの量が大幅に削減されます。作業がさまざまなスレッドで処理されるので、複数のジョブが必要になることはありません。スレッド化ジョブ・モデルを使用するサーバー・ジョブの例には、Lotus Domino®、HTTP サーバー、WebSphere® があります。

事前開始ジョブ・モデル
事前開始ジョブ・モデルには、通常、システムに入る要求の受話者としての役割を果たす 1 次ジョブがあります。このジョブは、一般的にデーモン・ジョブと呼ばれています。デーモン・ジョブは初期要求を処理してから、要求を適切な事前開始サーバー・ジョブに渡します。このジョブ・モデルの場合、事前開始ジョブの使用によって、必要なジョブの数を削減できます。要求が一度満たされると、事前開始サーバー・ジョブが次の要求を待機するからです。サーバー・ジョブは再利用されます。また、パフォーマンスの観点からすれば、事前開始ジョブがすでに実行されており、要求の処理を待機しています。事前開始ジョブ・モデルを使用するサーバー・ジョブの例には、 SQL サーバー、ホスト・サーバー、 Simple Mail Transfer Protocol (SMTP) があります。

注: ユーザー・コードを実行するジョブの場合、ジョブの再利用は基本的にありません (ほとんどのサーバー・ジョブと同様)。これは、ユーザー・コードがジョブの何か (リモート・コマンド・サーバーなど) を変更している可能性があるためです。

複数聴取ジョブ・モデル

複数聴取ジョブ・モデルでは、いくつかのサーバー・ジョブが開始されます。要求が入ってくると、要求を受け取るジョブがそのジョブ要求を扱い、次に使用可能なサーバー・ジョブが次の要求を待機します。サーバー・ジョブは要求を完了すると、接続をクローズして終了します。新しいサーバー・ジョブが開始し、サイクルが継続します。

このモデルを使用すると、事前開始ジョブ項目のことを気にする必要はありません。しかし、環境に固有のサブシステムを構成することができない場合があります。このモデルがデフォルト・サブシステムで実行しているためです。1 つの例外は、ファイル転送プロトコル (FTP) です。ファイル転送プロトコルを使用すると、ファイル転送プロトコル・サーバーが実行するサブシステムを構成することができます。FTP 作業の一部を 1 つのサブシステムで実行し、残りの作業を別のサブシステムで実行する機能はありません。また、パフォーマンスの観点からすれば、ジョブの開始と終了のコストを回避することはできません。これは、一度ジョブが実行されると、そのジョブが終了した後に別のジョブが開始するためです。しかし、接続が完了して次のジョブが開始されるとジョブが終了するため、次の要求が受け取られるときに、一般的には新しいジョブが稼働状態になるので、ジョブの開始と終了のコストは、サーバーに接続するのにかかる時間には影響しないはずです。

複数聴取ジョブ・モデルを使用するサーバー・ジョブの例には、FTP とライン・プリンター・デーモン (LPD) があります。

システムで実行するサーバー・ジョブのジョブ名の詳細については、サーバー・ジョブの表を参照してください。この表は、アクティブ・ジョブおよびそのジョブ・ログの検索を実行するためのサブシステムとジョブ名をまとめたものです。また、各サーバー・ジョブが使用するジョブ記述も示します。デフォルトでは、ほとんどのサーバー・ジョブは、ジョブ終了時にジョブ・ログを生成しません (LOG パラメーターが 4 0 *NOLIST に設定されているため)。つまり、ジョブ・ログは作成されません。ジョブ・ログに送信されるすべてのメッセージを含めてジョブ・ログを生成したい場合、LOG パラメーターの指定は 4 0 *SECLVL でなければなりません。

関連情報

サーバー・ジョブ・テーブル

システム・ジョブ:

システム・ジョブは、システム・リソースを制御したり、システム機能を実行するために、オペレーティング・システムによって作成されます。サーバーが起動されるか、または独立ディスク・プールがオンに変更されると、システム・ジョブが開始されます。 これらのジョブは、オペレーティング・システムの開始はもとより、またサブシステムの開始と終了、ジョブのスケジューリングにいたるさまざまなタスクを実行します。

システム始動ジョブ:

始動ジョブ は、IPL 時に実行されるシステム・ジョブです。これらは、オペレーティング・システム環境をセットアップして作業可能にするタスクを処理します。さまざまなシステム始動ジョブのリストを以下に示します。

Scpf (開始制御プログラム機能)

これが、システムを開始したときに中心的な役割をするジョブです。Scpf は一連の Qsysarb ジョブを開始しますが、Qsysarb3 はそれ以外のほとんどのシステム・ジョブ (Qlus を除く) を開始し

て、システムを使用可能な状態に戻します。このジョブはシステムが開始した後もアクティブ状態を継続し、低優先順位および実行時間が長くなる可能性のあるシステム機能を実行する環境を提供します。Scpf は電源遮断 (Pwrdwnsys) 処理中も実行し、マシン処理を終了させるジョブでもあります。

Qwcbtclnup (ジョブ・テーブル・クリーンアップ)
このジョブは、システムの始動時にジョブ構造体が使用可能であることを確認するために使用されます。 このジョブはシステム始動が終わる前に処理を完了しますが、終結処理 (クリーンアップ) するジョブ構造体がたくさんある場合には、システム開始後も実行を継続することができます。このシステム・ジョブは処理が完了すると終了します。

Qlpsvr (ソフトウェア使用許諾契約の同意)
オンライン・ソフトウェア使用許諾契約の同意が必要な場合に、 IPL 中にこのジョブが自動的に開始します。すべての使用許諾契約について同意または拒否が完了すると、このジョブは終了します。

システム・アービター:

SCPF システム・ジョブによって開始されるシステム・アービター (QSYSARB および QSYSARB2 から QSYSARB5) は、優先順位の高い機能を実行するための環境を提供します。これらは、サブシステムを開始および停止し、システムの状態 (制限状態など) を追跡します。

システム・アービター (ジョブ名 QSYSARB および QSYSARB2 から QSYSARB5 で識別される) は、オペレーティング・システム内で優先順位が最も高い、中心となるジョブです。各システム・アービターは、ただちに処理する必要があり、しかも複数のジョブよりも単一のジョブで扱う方が効率のよいシステム全体に関連するイベントに対処します。

システム・アービター (QSYSARB) は、IPL 時に論理装置サービス (QLUS) ジョブを開始します。システム・アービターは、システムが終了するまでアクティブのままです。

システム・アービターのリストを以下に示します。

Qsysarb (システム・アービター)
システム・アービターは高優先順位機能を実行する環境を提供します。システム・リソースを取り扱い、システムの状態に関する情報を最新に保ちます。システム・アービターは、ただちに処理する必要があり、しかも複数のジョブよりも単一のジョブで扱う方が効率のよいシステム全体に関連するイベントに対処します。 Qsysarb、Qtaparb (テープ・アービター)、および Qcmnarbxx (通信アービター) は、通信要求、装置ロック、回線、制御装置、および装置構成の処理、および他のシステム全体のリソースの処理を実行します。

Qsysarb2 (システム・アービター 2)
このジョブは、テープ・リソースの管理、コマンド処理のためのコマンド分析プログラム・スペースの処理、およびオペレーティング・システムに関する他のシステム全体の処理を実行します。

Qsysarb3 (システム・アービター 3)
このジョブはシステム上でのジョブ構造体の作成と保守を実行します。ジョブ開始に一時的なまたは永続的なジョブ構造体が必要である場合は必ず、要求が Qsysarb3 によって処理されます。 Qsysarb3 は多くのシステム・ジョブの開始および終了も実行します。

Qsysarb4 (システム・アービター 4)
このジョブはサブシステムの開始と終了を実行します。初期電源遮断 (Pwrdwnsys) 処理も含まれます。

Qsysarb5 (システム・アービター 5)
このジョブはマシン・イベントの処理を実行します。これには、補助電源、システム補助記憶域プール (ASP) および記憶域しきい値、およびロック・テーブル限界値のサポートのためのさまざまなイベントの処理が含まれます。通常は、マシン・イベントが処理され、対応する CPF メッセージが Qsysopr および Qhst に送信されます。

システム通信ジョブ:

このトピックでは、システム通信ジョブのリストを示します。

Qlus (論理装置サービス)
Qlus は論理装置 (通信装置とも呼ばれる) のイベント処理を行います。Qlus は正しい通信サブシステムへの装置の割り振りも行います。

Qcmnarbxx (通信アービター)
通信アービターは Qsysarb (システム・アービター) および Qtaparb (テープ・アービター) とともに、通信装置だけでなくすべての種類の装置の作業も処理します。この作業には、通信の接続、切断、装置ロック、およびエラー回復処理が含まれます。再始動時に通信アービター・ジョブ (QCMNARB) システム値は開始済みの通信アービター・ジョブの数を判別します。シングル・プロセッサー・システムでは最低で 3 つの通信アービターが開始されます。

Qsyscomm1 (システム通信)
このジョブはいくつかの通信および入出力 (I/O) アクティビティーを実行します。

Q400filsvr (リモート・ファイル・システム通信)
このジョブは、これらのリモート・ファイル・システムに対して共通プログラミング・インターフェース通信 (APPN または APPC) を実行します。

データベース・ジョブ:

この情報では、データベース・ジョブのリストを示します。

Qdbfstccol (データベース・ファイル統計収集)
このジョブはデータベース・ファイル統計を収集します。これらの統計は、正しいデータベース QUERY の最適化に対して決定的です。

Qdbsrvxr (データベース相互参照) および Qdbx###xr (独立ディスク・プール・グループ ### に対応)
このジョブは Qsys にある各ファイル・レベル・システム相互参照ファイルを保守します。これらのファイルには、システム内のデータベース・ファイルおよび SQL 情報に関する相互参照情報が含まれています。これらのファイルはライブラリー Qsys にあり、接頭部 Qadb で始まるものです。保守を必要とする主なファイルは Qadbxref という相互参照ファイルです。このファイルにはシステムにある物理データベース、論理データベース、DDM、別名ファイルのそれぞれに関するレコードが含まれています。 Qdbsrvxr は、ファイルが作成、変更、削除、復元、名前変更、またはその所有者が変更されたときにアクティブになります。

Qdbsrvxr2 (データベース相互参照 2) および Qdbx###xr2 (独立ディスク・プール・グループ ### に対応)
このジョブは 2 つのフィールド・レベル相互参照ファイルを保守します。ライブラリー Qsys にある Qadbifld はフィールド相互参照ファイルです。ライブラリー Qsys にある Qadbkfld はキー・フィールド相互参照ファイルです。 Qdbsrvxr2 は、ファイルが作成、変更、削除された時点でアクティブになります。

Qdbsrv01 (データベース・サーバー) および Qdbs###v01 (独立ディスク・プール・グループ ### に対応)
このジョブは、データベース保守タスク・ディスパッチャーと見ることができます。システム上のデータベース・サーバー・ジョブの数は、 1 にプロセッサーの数の二倍を加えたもの、または 1

に ASP の数の二倍を加えたもののうち、いずれか大きいほうの数です。開始される最小の数は 5 です。 Qsbsrv01 は、作業を他のジョブに割り振るメインのシステム・ジョブです。通常、データベース・ファイルを含むライブラリーが復元された直後に、Qdbsrv01 のアクティブ状態は最も高くなります。 このジョブには以下の機能が含まれます。

- 新しいアクセス・パスが復元されたことを、システム管理のアクセス・パス保護 (SMAPP) ライセンス内部コード (LIC) にシグナル通知します。すると、SMAPP はこれらのアクセス・パスを保護する必要があるかどうかを判断します。
- アクセス・パスが復元されなかったため再作成が必要なアクセス・パスのリストを準備します。

残りのデータベース・サーバー・ジョブについて、始めの半分の処理は高優先順位要求であり、残りの半分の処理は低優先順位要求です。 (例: Qdbsrv02 から Qdbsrv05 は高優先順位、Qdbsrv06 から Qdbsrv09 は低優先順位です。)

Qdbsrvxx (データベース・サーバー、高優先度) および Qdbs###vxx (独立ディスク・プール・グループ ### に対応)

これらのジョブはシステムのジャーナルおよびコミットメント制御保守を実行するので、高速または短時間実行作業と見なされます。

Qdbsrvxx (データベース・サーバー、低優先度) および Qdbs###vxx (独立ディスク・プール・グループ ### に対応)

これらのジョブはユーザー・データ・ファイル上でアクセス・パス保守を実行します。通常これらのジョブは非アクティブですが、特定の場合にアクセス・パス再作成の実行がアクティブになる場合があります。これらのジョブがアクティブ状態になる理由には次のいくつかのものがあります。

- アクセス・パスとともに保存されなかったデータベース・ファイルを復元した。
- 基になっている物理ファイルとは別に論理ファイルを復元した。
- Rgzpfm コマンドを処理中に取り消した。
- 索引内に見つかった損傷のために索引が無効になった。
- 相互参照を完了するポスト iServer インストール・アクティビティーまたはその他の DB アップグレード・アクティビティー。
- 制約の検証。

Qqqtemp1 および Qqqtemp2 (データベース並列性)

データベース並列性システム・ジョブは、DB2® マルチシステムに対する非同期のデータベース処理を実行します。分散しているファイルを照会する場合、このジョブを用いると、あるいくつかのタスクが並列的に処理されて、照会の速度が上がります。

他のシステム・ジョブ:

ここでは、他の種類のシステム・ジョブのリストを示します。

Qalert (警報マネージャー)

このジョブは、警報を処理するのに必要なタスクを実行します。これには、他のシステムから受け取った警報の処理、そのシステム自体で生成された警報の処理、および制御の範囲の維持などのアクティビティーが含まれます。

Qdcpobjx (圧縮解除システム・オブジェクト)

このジョブは、必要に応じて、新たにインストールされたオペレーティング・システム・オブジェクトを圧縮解除します。これらのジョブを実行するにあたって、記憶域要件があります。システムで使用可能な記憶域が一定の限界より下であれば、これらのジョブは終了します。圧縮解除システム・オブジェクト・ジョブの数は、プロセッサーの数に 1 を加えたものです。

Qfilesys1 (ファイル・システム)
このジョブは統合ファイル・システムのバックグラウンド・プロセスをサポートします。ファイルに加えられる変更は確実に記憶域に書き出され、一般ファイル・システム・クリーンアップ・アクティビティーもいくつか実行されます。

Qjobscd (ジョブ・スケジュール)
このジョブはシステムのジョブ・スケジュール機能を制御します。Qjobscd はジョブ・スケジュール項目とスケジュール済みジョブのタイマーをモニターします。

Qli###cl (独立ディスク・プール・グループ ### に対応する。(ライブラリー・クリーンアップ))
このジョブは独立ディスク・プール上のライブラリーをクリーンアップします。

Qli###rp (独立ディスク・プール・グループ ### に対応する。(オブジェクト・クリーンアップ))
このジョブは独立ディスク・プール・ライブラリー上の置換オブジェクトをクリーンアップします。

Qlur (LU 6.2 再同期)
Qlur は 2 フェーズ・コミット再同期処理を行います。

Qpfradj (パフォーマンス調整)
このジョブは、記憶域プール・サイズおよびアクティビティー・レベルへの変更を管理します。記憶域プールの変更要求は、すべてこのジョブが処理します。さらに、記憶域プールおよびアクティビティー・レベルの自動調整 (Qpfradj) システム値が 2 または 3 に設定されている場合には、このジョブは記憶域プールのサイズとアクティビティー・レベルを動的に変更してシステム・パフォーマンスの向上を計ります。

Qsplmaint (システム・スプール・メンテナンス) および Qspmn##### (独立ディスク・プール・グループ ##### に対応)
このジョブは以下のシステム・スプール機能を実行します。

- IPL 後または独立ディスク・プール・グループがオンに変更された後にスプール・ファイルをクリーンアップする
- ユーザー出力待ち行列の障害のためにサブシステム補助記憶域プールまたは基本ユーザー補助記憶域プールに取り残されたスプール・ファイルを、ライブラリー QRCL の出力待ち行列 QSPRCLOUTQ に移動する
- 削除されたスプール・ファイルのデータと属性が保持されているスプール・データベース・メンバーをクリアする
- 未使用のプリンター出力記憶域の自動クリーンアップ (QRCLSPLSTG) システム値に指定された時間内に再利用されなかったスプール・データベース・メンバーを削除する。

Qsppf##### (独立ディスク・プール・グループ ##### に対応。(システム・スプール PRTQ 更新プログラム))
このジョブは特定の独立ディスク・プール・グループに対してスプールされたファイル操作を実行します。

Qtaparb (磁気テープ装置)
このジョブは装置ロックおよびエラー回復処理を含む磁気テープ装置に関連した作業を処理します。

Qnwharbxx
これらのシステム・ジョブは、Network Server Host Adapter (NWSH) 装置に関連したイベントを処理します。現行 IPL 時に、これらのジョブの少なくとも 1 つが常に開始されます。

Qwcpjobs
このジョブは、永続的なジョブ構造体のバックグラウンド・クリーンアップを処理します。

Qwctjobs

このジョブは、一時的なジョブ構造体のバックグラウンド・クリーンアップを処理します。

ジョブ・スケジューリング・オプション

ジョブ・スケジュール機能では、System i バッチ・ジョブの時間依存スケジューリングが可能です。 特定の時刻にジョブ待ち行列からジョブが解放されるようスケジュールすることも、ジョブ・スケジュール項目を使用して指定の時刻にジョブをジョブ待ち行列に自動的に投入することもできます。ジョブ・スケジューリングにより、バッチ・ジョブをジョブ待ち行列に投入する日時、またはジョブ待ち行列から開始可能になる日時を制御できます。この柔軟な機能は、システム上で作業負荷のバランスを取る際に役立ちます。

例えば、ジョブ・スケジューリングを使用して、会議通知、給与計算、または週次および月次レポートを自分のスケジュールからシステムのスケジュールに繰り返し送信するという反復タスクを代行することができます。バッチ・ジョブは、4 通りの方法でスケジュールできます。

マネージメント・セントラル・スケジューラー

System i ナビゲーター には、ジョブを処理するタイミングを編成するための統合されたスケジューラーであるマネージメント・セントラル・スケジューラーが備えられています。 タスクをすぐに実行するか、それとも後で実行するかを選択することができます。マネージメント・セントラル・スケジューラーを使用すると、マネージメント・セントラル内のほとんどのタスクをスケジュールできます。

「マネージメント・セントラル・スケジューラー (Management Central Scheduler)」ウィンドウは、「System i ナビゲーター」ウィンドウで「**スケジュール**」ボタンが表示されているときにはいつでも使用できます。

注: マネージメント・セントラル・サーバーに Advanced Job Scheduler がインストールされている場合には、「**スケジュール**」ボタンを押すとマネージメント・セントラル・スケジューラーではなく Advanced Job Scheduler が開始されます。

関連タスク

149 ページの『マネージメント・セントラル・スケジューラーを使用したジョブのスケジュール』
プラグイン Advanced Job Scheduler をインストールしていない場合は、マネージメント・セントラル・スケジューラーを使用してジョブをスケジュールできます。

Advanced Job Scheduler

IBM® Advanced Job Scheduler for i5/OS (5761-JS1) ライセンス・プログラムは、1 日 24 時間、週 7 日の不在ジョブ処理が可能な優れたスケジューラーです。このスケジューリング・ツールは、マネージメント・セントラル・スケジューラーよりも多くのカレンダー機能と、スケジュールされたイベントに対するさらに拡張された制御を提供します。ジョブ完了履歴を表示したり、ジョブの状況の通知を管理することもできます。

ネットワーク内の複数のシステムでジョブをする場合は、製品を各システムにインストールする必要があります。 System i ナビゲーター (およびマネージメント・セントラル) の Advanced Job Scheduler を使用する場合、Advanced Job Scheduler がインストールされているシステムからクライアント・プラグインをインストールする必要があります。

ただし、Advanced Job Scheduler ライセンス・プログラムをマネージメント・セントラル・ネットワーク内の各エンドポイント・システムにインストールする必要はありません。 Advanced Job Scheduler をセントラル・システムにインストールすると、エンドポイント・システム上に定義したジョブまたはタスクが、必要なジョブ情報をセントラル・システムから集めるからです。 すべてのジョブ定義情報はセントラル・システムにセットアップしなければなりません。

ネットワーク内の複数のシステムに Advanced Job Scheduler がローカルにインストールされている場合は、マネージメント・セントラル・ネットワークの外でタスクをスケジュールすることになります。System i ナビゲーターの「**ユーザー接続**」で、「**実行管理機能**」を展開すると、ローカル・システムの Advanced Job Scheduler にアクセスします。

注: 注文情報については、 Job Scheduler for i5/OS  Web サイト (英文) を参照してください。

ジョブ・スケジュール項目

システムにマネージメント・セントラル・スケジューラーまたは Advanced Job Scheduler がない場合、文字ベース・インターフェースからアクセスできるジョブ・スケジュール項目を使用してジョブをスケジュールできます。この方法を使用すると、ジョブを繰り返し、または 1 度だけ実行するようにスケジュールできます。

ジョブ・スケジュール項目は永続オブジェクトにおける項目で、スケジュールされたジョブのようにジョブ待ち行列上に留まらないので、ジョブ待ち行列の消去時にジョブ・スケジュール項目が失われることはありません。さらに、ジョブ・スケジュール・オブジェクトを保管して復元することもできます。この方法により、ジョブ・スケジューリング情報をバックアップすることが可能になります。

規則的な間隔でジョブを処理したい場合、ジョブ用のジョブ・スケジュール項目を作成します。ジョブ・スケジュール項目には、ジョブの投入に必要な情報およびそのスケジューリング情報すべてが含まれます。オブジェクト内の各項目は、ご自分が提供するジョブ名、およびシステムによって割り当てられる 6 桁の項目番号によって一意的に識別されます。同じジョブ名と項目番号の組み合わせを持つ項目が 2 つあることはありません。

さらにジョブ・スケジュール項目には、システムが特定の状態で項目を管理するために使用する情報も含まれます。ジョブを定義するこの情報は、ジョブ投入 (SBMJOB) コマンドで指定されるパラメーターと類似していて、ジョブ名、ジョブ記述、ジョブ待ち行列、ユーザー・プロファイル、およびメッセージ待ち行列が含まれます。ジョブ・スケジュール項目から投入されるジョブのローカル・データ域 (LDA) は、ジョブ開始時にはブランクです。

すべてのジョブ・スケジュール項目は、ジョブ・スケジュール・オブジェクトに含まれます。ジョブ・スケジュール・オブジェクト QDFTJOBSCD は QUSRSYS ライブラリーにあり、オブジェクト・タイプ *JOBSCD を持っています。ジョブ・スケジュール・オブジェクトを作成、削除、名前変更、または複製することはできません。他のライブラリーに移動もできません。ジョブ・スケジュール・オブジェクトは、共通権限 *CHANGE を与えられて出荷されています。これは、ジョブ・スケジュール項目を追加、変更、保留、解放、および除去するために最低限必要な権限です。

注: マネージメント・セントラル・スケジューラーまたは Advanced Job Scheduler を使用しても、繰り返されるジョブをスケジュールできます。

関連概念

176 ページの『ジョブ・スケジュール項目の処理』

System i ナビゲーター の「ジョブのプロパティー」-「ジョブ待ち行列」ウィンドウ以外にも、文字ベース・インターフェースを使用してジョブ・スケジュール項目を直接変更することもできます。 以下に、ジョブ・スケジュール項目を処理するときに使用できる文字ベース・インターフェースの共通タスクのリストを示します。

例: ジョブ・スケジュール項目:

このトピックでは、ジョブ・スケジュール追加項目 (ADDJOBSCDE) コマンドを使用した例が提供されています。

月ごとのジョブのスケジュール: この例では、プログラム INVENTORY を毎月最終日 (大晦日の晩は除く) の午後 11:30 に実行するジョブを投入する方法を示しています。

```
ADDJOBSCDE JOB(MONTHEND)
CMD(CALL INVENTORY)
SCDDATE(*MONTHEND)
SCDTIME('23:30:00')
FRQ(*MONTHLY)
OMITDATE('12/31/05')
```

毎日のジョブのスケジュール: この例では、プログラム DAILYCLEAN を毎日午後 6:00 に実行するジョブの投入方法を示しています。このジョブは、ユーザー・プロファイル SOMEPGMR で実行されます。システムがダウンしているまたはその時点で制限状態にある場合には、このジョブは投入されません。

```
ADDJOBSCDE JOB(*JOBD)
CMD(CALL DAILYCLEAN)
SCDDAY(*ALL)
SCDTIME('18:00:00')
SCDDATE(*NONE)
USER(SOMEPGMR)
FRQ(*WEEKLY)
RCYACN(*NOSBM)
```

週ごとのジョブのスケジュール: この例では、現在時刻 12/17/05 に開始して、毎週プログラム PGM1 を実行するジョブを投入する方法を示しています。12/17/05 は土曜日ですので、ジョブは毎週土曜日に投入され、ユーザー・プロファイルで実行されます。

```
PGMR1. ADDJOBSCDE JOB(*JOBD)
CMD(CALL PGM1)
SCDDATE('12/17/05')
FRQ(*WEEKLY)
USER(PGMR1)
```

第 3 月曜日と水曜日ごとのジョブのスケジュール: この例では、第 3 月曜日と第 3 水曜日の午後 11:30 にプログラム PGM2 を実行するジョブを投入する方法を示しています。このジョブは、既に今月の第 3 月曜日または第 3 水曜日の午後 11:30 が過ぎているかどうかによって、次の第 3 月曜日か第 3 水曜日の 11:30 に投入されます。昨日が第 3 月曜日だった場合、今日は第 3 火曜日で、明日は第 3 水曜日の場合、ジョブは明日投入され、翌月までは再投入されません。

```
ADDJOBSCDE JOB(*JOBD)
CMD(CALL PGM2)
SCDDAY(*MON *WED) FRQ(*MONTHLY)
SCDDATE(*NONE)
RELDAYMON(3) SCDTIME('23:30:00')
```

第 1 月曜日と第 3 月曜日ごとのジョブのスケジュール: この例では、毎月の第 1 および第 3 月曜日の午前 9:00 にプログラム PAYROLL を実行するジョブを投入する方法を示しています。このジョブは、ユーザー・プロファイル PAYROLLMGR で実行されます。

```
ADDJOBSCDE JOB(PAYROLL)
CMD(CALL PAYROLL)
SCDDAY(*MON) FRQ(*MONTHLY)
SCDDATE(*NONE)
RELDAYMON(1 3) SCDTIME('09:00:00')
USER(PAYROLLMGR)
```

すべての平日ごとのジョブのスケジュール: この例では、すべての平日の午後 7:00 に PGM4 を実行するジョブを投入する方法を示しています。

```
ADDJOBSCDE JOB(*JOBD)
CMD(CALL PGM4)
SCDDAY(*MON *TUE *WED *THU *FRI)
SCDDATE(*NONE)
SCDTIME('19:00:00') FRQ(*WEEKLY)
```

ジョブ・スケジュール項目の保管: この例では、ジョブを一度投入してから項目を保管する方法を示しています。

```
ADDJOBSCDE JOB(*JOBD)
CMD(CALL SAVED)
FRQ(*ONCE)
SAVE(*YES)
```

ジョブ投入コマンド

この文字ベース・インターフェース・コマンドは、ジョブ待ち行列内でジョブが解放される時を制御します。これは、一度だけしか実行する必要がないジョブをスケジュールするための簡単な方法です。これにより、現行のジョブに対して定義されている多くのジョブ属性を使用できます。

ジョブを一度だけ実行するようにスケジュールする場合 (文字ベース・コマンド SBMJOB)、ジョブはスケジュールされた時刻にジョブ待ち行列から解放されます。 以下は、SBMJOB を使用してバッチ・ジョブをスケジュールする場合に実行されるシステム・タスクの要約です。

1. ジョブのスケジュールには、System i ナビゲーター インターフェース (**「基本操作」 → 「ジョブ」 → ジョブを右マウス・ボタン・クリック → 「プロパティー」 → 「ジョブ待ち行列」タブ**) または文字ベース・インターフェース (SBMJOB コマンドに SCDATE および SCDTIME パラメーターを指定) のいずれかを使用します。
2. ジョブは、これらのパラメーターによって指示された日時まで、スケジュール済み状態 (SCD 状況) のジョブ待ち行列上に残ります。
3. スケジュール時刻になると、ジョブはジョブ待ち行列から解放されます。ジョブの状況は、ジョブが保留状態 (SCDHLD) でなければ、スケジュール済み (SCD) から解放済み (RLS) に変わります。保留状態の場合には、スケジュール済みから保留 (HLD) に変わります。
4. ジョブは、ジョブ待ち行列上の他のジョブと同様に処理されます。
5. 通常の条件 (アクティブ・サブシステムおよび最大ジョブに割り振られているジョブ待ち行列がまだアクティブでないなど) が存在する場合に、ジョブは開始します。

注: この方法では、ジョブは即時にジョブ待ち行列に入れられるので、スケジュール日時前にジョブ待ち行列が消去されると、ジョブを解放することになります。

関連タスク

127 ページの『ジョブの 1 回の投入』
即時にまたはスケジュールした日時にジョブを 1 回実行する必要がある場合は、ジョブ投入 (SBMJOB) コマンドを使用します。 この方法では、ジョブはジョブ待ち行列に即時に入れられます。

131 ページの『バッチ・ジョブの投入』
一般に、バッチ・ジョブは、実行に特殊なシステム環境を必要とする優先順位の低いジョブであるため (夜間に実行するなど)、バッチ・ジョブ待ち行列に入れられます。ジョブ待ち行列内で、バッチ・ジョブはランタイム・スケジュールおよび優先順位を受け取ります。ジョブをバッチ・ジョブ待ち行列に投入するには、文字ベース・インターフェースおよび以下の 2 つのコマンドのいずれかを使用します。

ジョブ・スケジューラーの考慮事項

ジョブ・スケジューラー製品を選ぶときには、さまざまな特性について幅広く考慮することが必要です。どのジョブ・スケジューラーを使用するかを判断する際に考慮するとよい特性のリストを、以下に示します。

- **自動化ジョブ・スケジューリング**
  - ジョブをスケジュールする際の柔軟性
  - 1 日 24 時間、週 7 日の不在 (または在席) ジョブ処理が可能であり、設定したスケジュールを徹底して順守する
  - i5/OS オペレーティング・システムの自然な拡張である
  - ジョブを投入する方法、時刻、および場所が完全に制御されている
  - オブジェクト (物理ファイル内にファイルまたはレコードがある)、他のジョブのアクティブ状態または非アクティブ状態、あるいは回線、コントローラー、またはサブシステムの状況といった、幅広いジョブ依存関係
  - 会計カレンダーと祝祭日カレンダーを含む、完全なカレンダー機能
  - 1 日に複数の実行が可能である
- **システムおよびユーザー定義パラメーター**
  - 現在日付、投入日付、直前の日付、および現在時刻をアプリケーション・プログラムに渡せる
  - ユーザー定義パラメーター値を作成および変更でき、アプリケーション・プログラムに渡せる
- **ワークロード/履歴予測**
  - 来週、来月、または翌日実行するスケジュール済みジョブを予測できる
  - 実動要件を最適化できる
  - すべての Advanced Job Scheduler アクティビティーのヒストリカル・トラッキングおよびロギング
- **ネットワーク管理**
  - ネットワーク内のどの System i 製品上でもジョブをセットアップでき、そのジョブをネットワーク上の他の任意の System i 製品上で実行できる
  - 投入システム上で特定のジョブの完全なジョブ履歴を提供できる
  - グループおよび従属ジョブをネットワーク経由で投入できる
- **報告書の配布および管理**
  - Advanced Job Scheduler または i5/OS オペレーティング・システムによって生成されたすべての出力報告書の経路指定、モニター、および制御
  - スプール・ファイルにオプションのバナー・ページを付けて、複数の出力待ち行列またはリモート・システムに配布する
  - スプール出力を複製または i5/OS ネットワーク上の任意のユーザーに送信できる
- **セキュリティー**
  - 既存の i5/OS セキュリティーを Advanced Job Scheduler 内で使用できる
  - スケジュールされたジョブに関する情報をセットアップまたは変更する権限を組織内の誰が持つかを指定できる
  - Advanced Job Scheduler の個別の機能または特定のジョブのいずれかに対して権限を指定できる
- **グラフィカル・ユーザー・インターフェース**
  - ポイント・アンド・クリックで、ジョブをスケジュールできる
  - 複数のジョブを管理できる
  - 依存関係を維持できる
  - スケジューラー・アクティビティーおよびログ情報を追跡する
- **その他の主な特性**
  - 1 つのジョブで複数のコマンドを実行できる

– ジョブ LDA (ローカル・データ域) を定義できる
– コンソール・モニターを使用して制限状態にあるジョブを実行できる
– ジョブごとの最大実行時間をチェックできる
– サード・パーティー製のメッセージ・ベースのページング・システムに直接インターフェースを接続できる
– 各ジョブの完全なオンライン文書を提供できる
– カーソル移動に影響される拡張ヘルプ・テキストをすべての画面で表示できる

ジョブ・スケジューリングおよびシステム可用性

スケジュールされた時間になったときにシステムの電源が遮断されていたりシステムが制限状態にある場合には、ジョブをジョブ・スケジュール項目から投入できず、スケジュールされたジョブの状況を変更できません。しかし、システム IPL の後、または制限状態から出た後に、システムがこの状況に対処する方法を制御することができます。

ジョブ・スケジュール項目およびスケジュールされたジョブは、脱落したオカレンスが通常処理されたであろう順序で処理されます。脱落したジョブ・スケジュール項目およびスケジュールされたジョブが処理されている間に、他の正常な作業はシステムに入れられます。

- **ジョブ・スケジュール項目**: 項目の回復処置用に指定した値によって、各項目の処理方法を制御できます。この項目を使用してジョブを引き続き投入するか、ジョブを投入してジョブ待ち行列に保留するか、またはジョブを投入しないかを指定できます。ジョブを投入するよう要求すると、システムが利用できなかったときに脱落した投入がいくつあったとしても、各項目からジョブが 1 つだけ投入されます。
- **スケジュールされたジョブ**: システムが利用不可だった間にスケジュールされた時刻が過ぎたかどうかを、システムはチェックして判別します。スケジュールされたジョブで時刻を過ぎてしまったものがある場合、そのジョブの状況が更新されます。

ジョブ待ち行列

ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

ジョブ待ち行列のジョブが処理されるためには、ジョブ待ち行列からそのジョブを受け入れるアクティブなサブシステムがなければなりません。サブシステムが開始すると、作業を受け入れるよう構成されているジョブ待ち行列の割り振りが試行されます。ジョブ待ち行列のジョブが処理されるためには、その待ち行列が正しく割り振られる必要があります。そのため、サブシステムが複数のジョブ待ち行列のジョブを処理できるとしても、特定のジョブ待ち行列のジョブを処理できるのは一度に 1 つのサブシステムだけです。

サブシステムは優先順位に基づき、その各優先順位に構成されている制限内でジョブ待ち行列からジョブを選択します。ジョブにはそれぞれジョブ待ち行列の優先順位があり、ジョブがそのジョブ待ち行列に入れられているときにはそのジョブ・プロパティーによって管理されます。ジョブ待ち行列の基本セットがシステムに提供されています。さらに、必要に応じて追加のジョブ待ち行列を作成することができます。

注: 「ジョブ待ち行列のリストのオープン (QSPOLJBQ)」や「ジョブ待ち行列情報の検索 (QSPRJOBQ)」などの API を呼び出して、ジョブ待ち行列についての情報を取得することができます。

関連概念

207 ページの『ジョブ待ち行列の管理』
システムでの作業を管理する際、ジョブ待ち行列で待機中のジョブを扱う必要があることに気付く場合

があります。ジョブをすぐに実行することが必要な場合もあれば、ジョブを優先順位の低い状態で待ち行列に入れることもあります。あるいは、サブシステム上で保守を実行し、特定のサブシステムに関連付けられていない待ち行列にすべてのジョブを移動する必要がある場合もあります。

関連タスク

209 ページの『ジョブ待ち行列の消去』
ジョブ待ち行列を消去すると、待ち行列上のすべてのジョブが削除されます。これには、保留状態のすべてのジョブが含まれます。ジョブ待ち行列を消去するには、System i ナビゲーター ナビゲーターまたは文字ベース・インターフェースを使用できます。 実行中のジョブは、アクティブ・ジョブと見なされ、待ち行列上にはないので、影響を受けません。

210 ページの『ジョブ待ち行列の作成』
ジョブ待ち行列を作成するには、文字ベース・インターフェースを使用します。

210 ページの『ジョブ待ち行列の削除』
ジョブ待ち行列を削除するには、文字ベース・インターフェースを使用します。

212 ページの『ジョブ待ち行列の保留』
ジョブ待ち行列を保留状態にすると、現在ジョブ待ち行列で待機中のすべてのジョブの処理が妨げられます。ジョブ待ち行列を保留状態にしても、実行中のジョブには影響を与えません。待ち行列が保留中に追加のジョブをジョブ待ち行列に入れることができますが，それらは処理されません。

212 ページの『ジョブ待ち行列の解放』
ジョブ待ち行列を解放すると、ジョブ待ち行列が保留された結果として保留されたすべてのジョブも解放されます。ジョブ待ち行列が保留される前に個別のジョブが保留されていた場合は、そのジョブは解放されません。

関連情報

Work management APIs

番号付きリスト

番号付きリストの「番号」とは、ジョブ待ち行列に表示されるジョブの順番を指します。可用性、優先順位、および日時値は、ジョブ待ち行列上のジョブの順序を判別する上で役立ちます。

ジョブ番号は、ジョブ待ち行列でジョブが表示される場所を判別するためには使用されませんし、ジョブが実行されるタイミングにも影響を与えません。

可用性 ジョブ待ち行列上のジョブの状況を示します。可能な順番の値は、待機中、スケジュール済み、および保留です。

優先順位

ジョブ待ち行列上でのジョブの優先順位を表します。可能な優先順位値は 0 から 9 までで、0 が最高優先順位です。ジョブがスケジュールされたジョブの場合、ジョブ待ち行列でのジョブの順序について優先順位は関係しません。たとえば、2 つのジョブが 12:00:00 に実行されるようスケジュールされている場合、このジョブはジョブ・テーブル内のその位置に番号付けされます。

日時 ジョブの日時を示します。

- ジョブがスケジュールされている場合、この日時は、スケジュールされているジョブの実行日時を表しています。
- ジョブがスケジュールされていない場合、この日時は、ジョブがシステムに入れられた日時を表しています。

注: この日時は、特定のジョブ待ち行列に移動したジョブを正しい位置に手動で設定した日時となる場合もあります。

ジョブ待ち行列はどのように機能するか

ジョブ待ち行列は、ジョブ待ち行列項目によってサブシステムで割り振られます。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステムはその開始時に、待ち行列上のジョブを処理します。

サブシステム記述では、同時にアクティブにできるジョブ (バッチまたは対話式) の最大数を指定します。ジョブ待ち行列からアクティブにできるジョブの数は、ジョブ待ち行列項目に指定されます。

サブシステムの開始時に、ジョブ待ち行列上のすべてのジョブが必ずしも処理可能というわけではありません。スケジュールされたジョブは、ジョブ待ち行列に入れることができます。ジョブは、システム・オペレーターが解放するまで、待ち行列上で保留状態にしておくことができます。すべてのジョブが処理される前にサブシステムが終了した場合、ジョブは、サブシステムが再始動するまで、システム・オペレーターにより別のジョブ待ち行列に移動されるまで、システム・オペレーターにより削除されるまで、または別のサブシステムが同じジョブ待ち行列を割り振るまで、待ち行列上に残ります。

複数のサブシステム記述が同じジョブ待ち行列を参照できますが、一度に 1 つのアクティブなサブシステムだけしか、バッチ・ジョブのソースとしてジョブ待ち行列を使用できません。したがって、サブシステムが終了したがジョブがジョブ待ち行列上にある場合は、そのジョブ待ち行列を参照している別のサブシステムが開始して、そのジョブを処理することができます。別のサブシステムがすでに開始済みで、同じジョブ待ち行列を待機している場合、サブシステムは、使用可能になると、自動的にそのジョブ待ち行列を割り振ります。

関連概念

208 ページの『サブシステムは複数のジョブ待ち行列をどのように処理するか』
サブシステムが複数のジョブ待ち行列をどのように処理するかを明らかにするために、以下のシナリオを考慮してください。

関連タスク

211 ページの『ジョブ待ち行列が割り振られているサブシステムの判別』
どのサブシステムがジョブ待ち行列を割り振ったかを判別するには、System i ナビゲーター インターフェースまたは文字ベース・インターフェースを使用します。 サブシステムがアクティブになっているジョブ待ち行列は削除できないため、これはジョブ行列を削除する必要がある場合に役立ちます。

210 ページの『ジョブ待ち行列の作成』
ジョブ待ち行列を作成するには、文字ベース・インターフェースを使用します。

208 ページの『サブシステムへのジョブ待ち行列の割り当て』
ジョブ待ち行列項目をサブシステム記述に割り当てるには、文字ベース・インターフェースを使用します。

ジョブ待ち行列からジョブが取り出される方法

ジョブ待ち行列からジョブが選択され、開始される方法を決めるさまざまな要素があります。

サブシステムごとの最大アクティブ・ジョブ数 (Maximum active jobs for subsystems)
これは、サブシステムで実行できるジョブの最大数を表します。この制限に達すると、そのサブシステムではそれ以上ジョブは開始されません。

ジョブ待ち行列ごとの最大アクティブ・ジョブ数 (Maximum active jobs for job queues)
これは、サブシステム内で同時に実行できるジョブ待ち行列のジョブの最大数を表します。この制限に達すると、そのジョブ待ち行列からそれ以上ジョブは開始されません。

ジョブ待ち行列の優先順位 (Priority on job queue)
実行を待っているジョブは、ジョブ待ち行列優先順位に基づいて選択されます。サブシステムは優

先順位の高いジョブ (ジョブ待ち行列の優先順位は 0 から 9 までで、 0 が最も高い優先順位です) から先に実行しようとしますが、優先順位に基づいて実行されているジョブの数が優先順位ごとの「最大アクティブ・ジョブ数」に達すると、次の優先順位のものが処理されます。 (同じ優先順位のジョブがジョブ待ち行列に入れられる場合、最初のジョブがまず実行され、次に 2 番目のものというように順番に処理されます。)

順序　サブシステム記述のジョブ待ち行列項目に順序を指定します。順序番号は、サブシステムがジョブ待ち行列を処理する順番を定義します。サブシステムは、最初に順序番号が最小のジョブ待ち行列のジョブを処理します。ジョブ待ち行列にジョブがなくなる場合、ジョブ待ち行列の 1 つが最大値に達する場合、サブシステムは次に高い順序番号のジョブ待ち行列を処理します。

関連タスク

214 ページの『ジョブのジョブ待ち行列への配置』
既存のジョブをある待ち行列から別の待ち行列に移動するか、新規ジョブを投入することによって、ジョブはジョブ待ち行列に入れられます。待ち行列間でジョブを移動するには、System i ナビゲーターを使用します。 新規ジョブを投入するには、文字ベース・インターフェースを使用します。

213 ページの『別のジョブ待ち行列へのジョブの移動』
さまざまな理由により、ジョブを別の待ち行列に移動したい場合があります。たとえば、ジョブが長く実行されているため、ジョブが待ち行列内でバックログになることがあります。おそらく、ジョブのスケジュールされた実行時間が高位の優先順位を持つ新しいジョブと競合しているのかもしれません。このような状況を管理する 1 つの方法は、待機中のジョブを使用率のそれほど高くない別の待ち行列に移動することです。

209 ページの『ジョブ待ち行列内で同時に実行するジョブ数の変更』
QBASE サブシステムには、QBATCH ジョブ待ち行列用のジョブ待ち行列項目が付属します。この項目では、一度に 1 つのバッチ・ジョブしか実行できません。そのジョブ待ち行列から複数のバッチ・ジョブを同時に実行したい場合は、ジョブ待ち行列項目を変更する必要があります。

ジョブ待ち行列項目

ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列項目には、ジョブ待ち行列の処理方法を制御する 5 つのパラメーターがあります。

サブシステム記述 (SBSD)
ジョブ待ち行列項目を追加するサブシステム記述の名前およびライブラリーです。

ジョブ待ち行列 (JOBQ)
サブシステムによって開始されるバッチ・ジョブの送信元であるジョブ待ち行列の名前およびライブラリーを指定します。

アクティブ・ジョブの最大数 (MAXACT)
このジョブ待ち行列から同時にアクティブ状態にできるジョブの最大数を指定します。

順序番号 (SEQNBR)
このジョブ待ち行列の順序番号を指定します。これは，ジョブ待ち行列が処理される順序を決定するためにサブシステムによって使用されます。

最大アクティブ優先順位 1 (9 まで) (MAXPTYx)
指定されたジョブ優先順位レベルで開始できるジョブの数を指定します。

関連タスク

184 ページの『ジョブ待ち行列項目の追加』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列から開始されるジョブはバッチ・ジョブです。ジョブ待ち行列項目は、文字ベース・インターフェースを使用して追加します。

189 ページの『ジョブ待ち行列項目の変更』
指定されたサブシステム記述内の既存のジョブ待ち行列項目は変更可能です。このコマンドは、サブシステムがアクティブでも非アクティブでも発行できます。サブシステム内のジョブ待ち行列項目を変更するには、文字ベース・インターフェースを使用します。

192 ページの『ジョブ待ち行列項目の除去』
文字ベース・インターフェースを使用して、ジョブ待ち行列項目をサブシステム記述から除去できます。ジョブ待ち行列項目がサブシステム記述から除去される時に、そのジョブ待ち行列上のジョブは待ち行列に残っています。現在アクティブ状態のいずれかのジョブがジョブ待ち行列から開始された場合には、ジョブ待ち行列項目を除去することはできません。

209 ページの『ジョブ待ち行列内で同時に実行するジョブ数の変更』
QBASE サブシステムには、QBATCH ジョブ待ち行列用のジョブ待ち行列項目が付属します。この項目では、一度に 1 つのバッチ・ジョブしか実行できません。そのジョブ待ち行列から複数のバッチ・ジョブを同時に実行したい場合は、ジョブ待ち行列項目を変更する必要があります。

ジョブ待ち行列がサブシステムに割り振られる方法

1 つのジョブ待ち行列を複数のサブシステムに関連付けることができますが、一度に割り振ることができるのは 1 つのサブシステムに対してだけです。サブシステムが開始されると、サブシステム・モニターはサブシステムのジョブ待ち行列項目で定義されている各ジョブ待ち行列を割り振ろうとします。

ジョブ待ち行列が別のサブシステムによって既に割り振られている場合には、2 番目のサブシステムがその待ち行列を割り振るには最初のサブシステムがジョブを終了して割り振り解除する必要があります。2 番目のサブシステムが開始された後、そのサブシステムに割り振られたジョブ待ち行列が使用可能になると割り振ります。

サブシステムの開始時にジョブ待ち行列が存在しないと、以下のいずれかが生じる際にジョブ待ち行列がサブシステムに割り振られます。

- ジョブ待ち行列が作成される。
- ジョブ待ち行列が、サブシステムで定義された名前に名前変更される。
- ジョブ待ち行列が別のライブラリーに移動され、その結果生じる修飾名がサブシステム記述内の名前と一致する。
- ジョブ待ち行列を含むライブラリーが名前変更され、その結果生じる修飾名がサブシステム記述内の名前と一致する。

複数のジョブ待ち行列

多くの場合、1 つのアクティブ・ジョブのデフォルトとともに QBATCH を唯一のジョブ待ち行列として使用すれば、必要に十分応えられます。これでは不十分な場合、一部は通常の勤務時間中アクティブ状態にし、一部は特殊用途に当て、一部は通常の勤務時間後アクティブ状態にするというように複数のジョブ待ち行列を持ちたいと思う場合があります。

たとえば、異なるジョブ待ち行列を次のような用途に指定することができます。

同時にアクティブ状態になっているジョブの数を制御するための、長期実行ジョブ用
これらのジョブでは、他のバッチ・ジョブよりも低い優先順位を使用することもできます。

通常の勤務時間中に実行するのは適切でない終夜ジョブ用
たとえば、大きなデータベース・ファイルに対して物理ファイル・メンバー再編成 (RGZPFM) コマンドを実行するには、そのファイルの排他ロックが必要です。つまり、この操作が実行されてい

る間、他のユーザーはファイルにアクセスできません。さらに、この操作には長い時間がかかることがあります。このジョブは、勤務時間外に実行するジョブ用のジョブ待ち行列に入れた方がより効率的です。

優先順位の高いジョブ

優先順位の高いすべての作業が送られるジョブ待ち行列を持つこともできます。これにより、この種の作業が迅速に完了し、優先順位の低いジョブによって遅らされることのないことが保証されます。

ディスケットやテープなどの特定のリソース要件に向けられるジョブ

そのようなジョブ待ち行列では、一度に 1 つのジョブだけでリソースを使用できるように、サブシステム記述のジョブ待ち行列項目に 1 という値を持つ MAXACT パラメーターが必要になります。

たとえば、テープが複数のジョブに使用される場合は、テープを使用するすべてのジョブが単一のジョブ待ち行列に入れられます。その上で、そのジョブ待ち行列から一度に 1 つずつジョブを選択します。これにより、2 つのジョブが同時に同じ装置を要求して競合が発生することがないよう保証されます。このような競合が起こった場合は、一方のジョブは割り振りエラーで打ち切られることになります。

注: テープ出力はスプールできません。

プログラマー作業

プログラマー作業、または実動作業の実行中は保留されてもよいタイプの作業を処理するジョブ待ち行列を備えることもできます。

一連のジョブの順次実行

アプリケーションで、1 つのジョブを他のジョブの完了に従属させることができます。一度にジョブを 1 つずつ選択および実行するジョブ待ち行列にそのようなジョブを入れる場合は、このようにするとそれらのジョブの実行順序を確保できます。

あるジョブがファイルの排他的制御を必要とする場合、そのジョブをジョブ待ち行列に入れるのを、サーバー上でアクティブになっているのがそのジョブ待ち行列だけのとき (夜間または週末など) にすることができます。

複数のジョブ待ち行列を使用する場合は、様々なジョブ待ち行列の制御が主な考慮事項になります。通常、制御の対象にする必要のあるのは次の各事項です。

- 存在するジョブ待ち行列の数。
- 特定のサブシステムの中で同時にアクティブ状態にあるジョブ待ち行列の数。
- ある時点に特定のジョブ待ち行列から選択できるアクティブ・ジョブの数。
- ある時点にサブシステムの中でアクティブ状態であり得るジョブの数。

ジョブを複数のジョブ待ち行列から取り出す方法

サブシステムは、ジョブ待ち行列からのジョブを順序番号に基づいて処理します。サブシステムは複数のジョブ待ち行列項目を持つことができ、そのため複数のジョブ待ち行列を割り振ることができます。

待ち行列からのジョブの最大数は、ジョブ待ち行列項目追加 (ADDJOBQE) コマンドまたはジョブ待ち行列項目変更 (CHGJOBQE) コマンド上の最大アクティブ・ジョブ MAXACT パラメーターで指定します。 さらに、各優先順位でアクティブにできるジョブの数は、最大アクティブ優先順位 MAXACTx パラメーターを使用して制御できます。たとえば、MAXACT=10、MAXACT5=2、および優先順位レベル 5 でジョブ待ち行列上に 3 つのジョブがある場合、そのうちの 2 つだけが特定の時点でアクティブになることができます。

サブシステムは、最初に順序番号が最小のジョブ待ち行列のジョブを処理します。ジョブ待ち行列上のすべてのジョブが処理された場合、または待ち行列からのジョブの最大数に達した場合、サブシステムは次に大きい順序を持つ待ち行列のジョブを処理します。

この手順は、サブシステムが選択可能なすべてのジョブ待ち行列項目を処理するか、またはサブシステムが、サブシステムで実行中または待機中にできるジョブの限度に達するまで続きます。実行中または待機中にできるジョブの数は、サブシステム記述内の最大アクティブ・ジョブ (MAXACT) パラメーターにより決定されます。 場合によってはこの手順はジョブ終了やジョブ転送などで中断されます。ジョブ待ち行列の作成、保留、または解除によっても、処理されるジョブ待ち行列の順序が変わることがあります。

関連タスク

214 ページの『ジョブのジョブ待ち行列への配置』
既存のジョブをある待ち行列から別の待ち行列に移動するか、新規ジョブを投入することによって、ジョブはジョブ待ち行列に入れられます。待ち行列間でジョブを移動するには、System i ナビゲーターを使用します。 新規ジョブを投入するには、文字ベース・インターフェースを使用します。

213 ページの『別のジョブ待ち行列へのジョブの移動』
さまざまな理由により、ジョブを別の待ち行列に移動したい場合があります。たとえば、ジョブが長く実行されているため、ジョブが待ち行列内でバックログになることがあります。おそらく、ジョブのスケジュールされた実行時間が高位の優先順位を持つ新しいジョブと競合しているのかもしれません。このような状況を管理する 1 つの方法は、待機中のジョブを使用率のそれほど高くない別の待ち行列に移動することです。

209 ページの『ジョブ待ち行列内で同時に実行するジョブ数の変更』
QBASE サブシステムには、QBATCH ジョブ待ち行列用のジョブ待ち行列項目が付属します。この項目では、一度に 1 つのバッチ・ジョブしか実行できません。そのジョブ待ち行列から複数のバッチ・ジョブを同時に実行したい場合は、ジョブ待ち行列項目を変更する必要があります。

ジョブ待ち行列のセキュリティー

ジョブ待ち行列に対する権限を一部の人 (ユーザー・プロファイル) に限ることによって、そのジョブ待ち行列のセキュリティー・レベルを維持することができます。一般的に、あるユーザーがジョブ待ち行列を制御する (たとえば、ジョブ待ち行列を保留または解放する) 権限を持つことができるようになるには、次の 3 つの方法があります。

- ユーザー・プロファイルの中で、ユーザーにスプール制御権を割り当てます (SPCAUT (*SPLCTL))。
- ユーザー・プロファイルの中でユーザーにジョブ制御権限を割り当て (SPCAUT (*JOBCTL))、ジョブ待ち行列はオペレーターが制御できる (OPRCTL (*YES)) ようにします。
- ユーザーが、ジョブ待ち行列に対して必要なオブジェクト権限を持ちます。必要なオブジェクト権限は、CRTJOBQ コマンドの AUTCHK パラメーターによって指定します。 *OWNER という値は、ジョブ待ち行列の所有者だけがそのジョブ待ち行列に対するオブジェクト権限によって権限を認可されることを示します。*DTAAUT という値は、ジョブ待ち行列に対する *CHANGE 権限を持つユーザーに、そのジョブ待ち行列を制御する権限が認可されることを示します。

注: *DTAAUT に必要な特定権限には、*READ、*ADD、および *DLT データ権限があります。

上記の 3 通りの権限認可方式が適用されるのはジョブ待ち行列に限られ、ジョブ待ち行列上のジョブには適用されません。ジョブがジョブ待ち行列上にあるかどうか、またはジョブが現に実行中であるかどうかにかかわらず、ジョブの制御に関する通常の権限規則が適用されます。

出力待ち行列

出力待ち行列とは、プリンター出力ファイル (スプールされるファイルとも呼ばれる) が処理され、プリンターに送信されるのを待機する領域です。プリンター出力は、システム、または印刷ファイルを使用するユーザーのいずれかによって作成されます。

印刷ファイルは、プリンター出力の属性のデフォルト値が設定されているテンプレートまたはガイドラインのようなものです。これは、プリンター出力のライフ・サイクルの始まりです。

印刷ファイルには、プリンター出力を送信する方法を指定する、出力待ち行列 (OUTQ) 属性およびプリンター (DEV) 属性が含まれます。デフォルト設定は通常 *JOB です。この設定では、プリンター出力を送信する方法は、出力待ち行列およびプリンターのジョブ属性によって判別されます。設定されている出力待ち行列およびプリンターのジョブ属性は、ジョブの作成時に入手される情報に基づいています。これは、ジョブを実行しているユーザーのユーザー・プロファイル、ジョブ記述、ワークステーション装置記述、および印刷装置記述 (QPRTDEV) システム値からの情報に基づいています。

プリンター出力が作成可能な状態になると、システムは印刷ファイルおよびジョブ属性を (この順序で) 検査して、プリンター出力を処理する出力待ち行列、およびシステムが使用するプリンターを確認します。ジョブを投入する際、または拡張処理をう回するためのジョブ実行時に、出力待ち行列 (OUTQ) およびプリンター (DEV) のパラメーターを変更できます。たとえば、ジョブの変更を開始してその効果がすぐに現れるようにする場合、印刷ファイル出力待ち行列を特定の待ち行列に設定し、プリンターをその印刷ファイルの特定のプリンターに設定できます。そのようにする際、プリンター出力は出力待ち行列および使用するプリンターを検出するために、ジョブ属性に入れられる必要はありません。特定の出力待ち行列が検出できない場合、プリンター出力は QGPL/QPRINT に送られます。プリンター出力を作成する方法の詳細については、「印刷装置プログラミング」の第 1 章を参照してください。

プリンター出力ファイルは、印刷および処理を待っている情報が入っているファイルです。プリンター出力ファイルは、他のプリンター出力と関連する、待ち行列上のプリンター出力の位置を定義する重要な属性を持っています。位置は、優先順位、状況、およびスケジュール属性によって定義されます。

出力待ち行列

出力待ち行列は、出力装置に書き出されるプリンター出力ファイルのリストを含むオブジェクトです。出力待ち行列は、プリンター出力が処理される順序、およびプリンター出力ファイルを変更するのに必要な権限を判別する重要な属性を持っています。

優先順位

処理を待っているプリンター出力は、優先順位に基づいて出力待ち行列に移動します (優先順位は 1 から 9 までで、 1 が最も高い優先順位です)。

状況　現在のプリンター出力の状況。「出力」プロパティー・ウィンドウの「汎用 (General)」ページから、この状況を表示できます。

スケジュール

スケジュール属性は、出力データの物理的な印刷をファイルが開始すべき時を通知します。

即時　プリンター出力ファイルがクローズしていない場合でさえ、即時に印刷します。

ファイル終了 (デフォルト)

プリンター出力ファイルがクローズするとすぐに印刷を開始します。

ジョブ終了

ジョブが終了すると、印刷を開始します。

プリンター出力ファイルが印刷可能な状態になった後、書き出しプログラム・ジョブ (出力待ち行列からプリンターに対してプリンター出力を処理するジョブ) は、プリンター出力ファイルからデータを取り出し、指定されたプリンターに送信します。

関連概念

215 ページの『出力待ち行列の管理』
出力待ち行列は、ジョブの終了時にプリンター出力を管理するために役立ちます。出力待ち行列を効率的に保守して印刷出力を順調に処理するための方法を理解することは大切です。

関連情報

経験報告: スプール・パフォーマンスの考慮点

印刷の基本

出力待ち行列の属性

出力待ち行列は、プリンター出力ファイル (スプール・ファイルとも呼ばれる) の処理方法と、出力待ち行列とその関連プリンター出力にアクションを実行する権限を持つユーザーを制御します。

システムで印刷する情報のほとんどはプリンター出力として作成されるので、権限のないユーザーが重要な機密データにアクセスしないようにするには、セキュリティーが必要となります。出力待ち行列やプリンター出力ファイルにアクセスして変更するには、検査の権限、データ権限、オペレーター制御、スプール制御、所有者権限のいずれかが必要です。出力待ち行列またはプリンター出力に任意のアクションを実行するには、以下の権限のいずれかが必要です。

検査の権限
待ち行列の所有者であるかデータ権限がなければなりません。

データの表示
この権限が *YES に設定されている場合、出力の表示、別のシステムへの移動や送信、およびプリンター出力のコピーなどのアクションを実行できます。

オペレーター制御
この属性が *YES に設定されていると、 *JOBCTL 特殊権限を持つユーザーは、保留、解放、および出力待ち行列からのプリンター出力の削除などのアクションを実行する権限が与えられます。プリンター出力、出力待ち行列、および書き出しプログラムに対する他のアクションも可能です。

スプール制御
ユーザーは、プリンター出力のすべての操作を実行できます。ユーザーが出力待ち行列に任意のアクションを実行するには、出力待ち行列が配置されているライブラリーに対する *EXECUTE 権限が必要です。

所有者 出力待ち行列を所有するユーザーは、プリンター出力を変更または削除できます。

注: 出力待ち行列に対するデフォルト権限は、*USE 共通権限です。データ表示権限は *NO に設定されています (だれもプリンター出力を表示できないことを意味します)。検査の権限は *OWNER です (したがって、出力待ち行列の所有者はプリンター出力を操作できます)。オペレーター制御は *YES に設定されています (*JOBCTL のユーザーはプリンター出力の保留、解放、および削除が可能です)。

i5/OS 権限について詳しくは、「Security Reference」の一連のトピックの『コマンドによって使用されるオブジェクトで必要な権限』を参照してください。

ファイルの順序

待ち行列上のファイルの順序 (SEQ) 属性は、プリンター出力が出力待ち行列から出て処理を受ける順序を判別します。

この属性には、以下の 2 つの値があります。

- *FIFO: 待ち行列は、ファイルごとの優先順位内で先入れ先出し法になります。つまり、新しくスプールされるファイルは、同じ優先順位の待ち行列上の他のすべての項目の後ろに置かれます。
- *JOBNBR : スプール・ファイルの待ち行列項目は、スプール・ファイルを作成したジョブのジョブ番号 (実際には、ジョブがシステムに入った日時が使用される) を使って優先順位に従ってソートされます。

注: 出力待ち行列のファイルの順序属性を変更できるのは、待ち行列上にプリンター出力ファイルがない場合だけです。

スプール・ファイル

スプーリングは、遅延処理または印刷用にデータを保管するシステム機能です。このデータはスプール・ファイルに保管されます。スプール・ファイルは、テープ・ファイルまたは他の装置ファイルと似た方法で機能します。スプール・ファイルにより、プリンターなどの外部接続装置を宛先とするデータを管理できます。

スプーリング機能は、サーバーのユーザーが入出力操作をより効率的に管理するのに役立ちます。サーバーは、出力スプーリングと入力スプーリングという 2 つのタイプのスプーリングをサポートします。出力スプーリングは、プリンター装置に使用できます。入力スプーリングは、データベース・ファイル入力に使用されます。

関連情報

スプール・ファイルと出力待ち行列

出力スプーリング:

出力スプーリングは、プリンターおよびディスケット装置のどちらにも使用できます。出力スプーリングは、ジョブ出力を直接プリンターまたはディスケット出力装置に送信するのではなく、ディスク・ストレージに送信します。出力スプーリングを使用すると、出力を生成するジョブは、出力装置の速度または可用性を考慮せずに処理を続けることができます。

さらに出力スプーリングを使用することにより、サーバーは、プリンターやディスケット装置などの複数の出力装置で効果的な方法で出力を生成できます。プリンターに向かうジョブの出力をディスク・ストレージに送信することにより、これを行えます。この処理によって、出力装置の可用性または速度に課されていた潜在的なジョブ制限以上のことが可能になります。

出力スプーリングの主要な要素は、以下のとおりです。

- **装置記述:** プリンター装置の記述。
- **スプール・ファイル** 出力装置で処理されるスプール出力レコードを含むファイル。
- **出力待ち行列:** スプール・ファイルの番号付きリスト。
- **書き出しプログラム:** ファイルを出力待ち行列から装置に送るプログラム。
- **アプリケーション・プログラム:** スプーリング属性として SPOOL(*YES) が指定されている装置ファイルを使用して、スプール・ファイルを作成する高水準言語プログラム。
- **装置ファイル:** 出力フォーマットの記述と、サーバーがスプール・ファイルを処理する方法を記述する属性のリスト。

出力スプーリング機能は、サーバーによって実行され、出力を作成するプログラムによる特別な操作を必要とはしません。ある装置ファイルがプログラムによってオープンされると、オペレーティング・システム

で、出力をスプール出力とすべきかどうかを決めます。スプーリングを指定する印刷装置ファイルがオープンされると、プログラムの出力が入っているスプール・ファイルが、サーバーの中の該当する出力待ち行列に入れられます。

スプール・ファイルは、印刷装置ファイルが開かれたとき、印刷装置ファイルが閉じられたとき、またはジョブが終了したときに、印刷用に使用可能になります。印刷装置書き出しプログラムがスプーリング・サブシステムの中で始動されて、レコードをプリンターに送ります。スプール・ファイルは出力待ち行列から選択されます。

スプーリング装置記述

各プリンターおよびディスケット装置ごとに装置記述を作成して、サーバーに対してその装置を定義しなければなりません。プリンターの装置記述は印刷装置記述作成 (CRTDEVPRT) コマンドを使用して作成し、ディスケット装置の装置記述はディスケット装置記述作成 (CRTDEVDKT) コマンドを使用して作成します。

スプール・ファイルのファイル指定変更

スプール・ファイルが最初に意図されていたものとは異なる出力装置に送信されるときに、ファイルの指定変更が生じます。ファイル指定変更には、異なるメディアを処理する場合 (ディスケット装置に送られたプリンター出力など)、または処理するのは同一タイプのメディアでも装置タイプが異なる場合 (4224 印刷装置に送られた 5219 印刷装置出力など) があります。

スプール・ファイルの新しい出力装置によっては、ファイルはもともと指定されていた装置で処理される場合と同じように処理されます。しかし、装置の違いにより、出力が異なった仕方で様式化されることもしばしばあります。これらの場合には、サーバーでは、書き出しプログラムのメッセージ待ち行列に照会メッセージを送ってユーザーに状況を通知し、ユーザーが印刷の続行を望むかどうかを指定できるようにします。

出力待ち行列とスプール・ファイル:

バッチおよび対話式ジョブによる処理を行うと、プリンターまたはディスケット装置などの出力装置で処理されるスプールされた出力レコードが生じます。こうした出力レコードは、処理されるまでスプール・ファイルに保管されます。1 つのジョブは複数のスプール・ファイルを持つことができます。

スプール・ファイルが作成されると、このファイルは出力待ち行列に置かれます。各出力待ち行列には、スプール・ファイルの番号付きリストが含まれます。1 つのジョブで、1 つ以上の出力待ち行列にスプール・ファイルを持つことができます。特定の出力待ち行列上のスプール・ファイルすべてには、装置、用紙タイプ、および行/インチなどの出力属性の共通セットがあります。出力待ち行列上の共通属性を使用すると、介入要求の量を減らし、装置スループットが増加します。

以下に、出力待ち行列作成 (CRTOUTQ) コマンドのパラメーターの一部およびそれらが指定する事柄をリストします。

- MAXPAGES: 開始時刻と終了時刻の間に印刷できる、ページ内の最大スプール・ファイル・サイズを指定します。
- AUTOSTRWTR: この出力待ち行列に対して自動的に開始される書き出しプログラムの数を指定します。
- DSPDTA: 出力待ち行列に対して特殊権限はないものの *USE 権限を持つユーザーが、独自のコンテンツではなくスプール・ファイルのコンテンツを表示、コピー、または送信できるかを指定します。DSPDTA に *OWNER を指定すると、ファイルの所有者または *SPLCTL 特殊権限を有するユーザーのみがファイルを表示、コピー、または送信できます。
- JOBSEP: 必要な場合に、出力の印刷時に各ジョブの出力の間に印刷するジョブ区切りページ数です。

- DTAQ: この出力待ち行列に関連付けられたデータ待ち行列。 指定すると、スプール・ファイルが待ち行列上で印刷可能状況になっても、項目はデータ待ち行列に送信されます。
- OPRCTL: ジョブ制御権限を持つユーザーが出力待ち行列を制御できるかどうかを指定します (たとえば、ユーザーが出力待ち行列を保留できる場合)。
- SEQ: スプール・ファイルが出力待ち行列上で保管される順序を制御します。
- AUTCHK: 出力待ち行列に対してどのタイプの権限を持っているユーザーが出力待ち行列上のスプール・ファイルを制御することができるかを指定します (たとえば、出力待ち行列上のスプール・ファイルをユーザーが保留できるようにします)。
- AUT: 共通権限です。 出力待ち行列に対してユーザーが有する制御を指定します。
- TEXT: テキスト記述です。 出力待ち行列について説明する、最大 50 文字のテキスト。

デフォルトのシステム出力待ち行列:

CL コマンドのデフォルトでは、スプールされたすべての出力のデフォルト出力待ち行列としてシステム・プリンターのデフォルト出力待ち行列が使用されます。システム・プリンターは QPRTDEV サーバー値によって定義されます。

スプール・ファイルが装置ファイルをオープンすることによって作成され、そのファイルに関する出力待ち行列が見つからないときは、システムはそのスプール・ファイルをライブラリー QGPL の中の出力待ち行列 QPRINT に入れようと試みます。 何らかの理由により、スプール・ファイルが出力待ち行列 QPRINT に入れられない場合はエラー・メッセージが送られ、出力はスプールされません。

以下の出力待ち行列が提供されます。
- **QDKT**: デフォルトのディスケット出力待ち行列
- **QPRINT**: デフォルト・プリンター出力待ち行列
- **QPRINTS**: 特殊用紙用のプリンター出力待ち行列
- **QPRINT2**: 2 部用紙用のプリンター出力待ち行列

スプール書き出しプログラム:

書き出しプログラムは、スプール・ファイルを出力待ち行列から出力装置に取り出す i5/OS プログラムです。 特定の出力待ち行列に入れられているスプール・ファイルは、書き出しプログラムが出力待ち行列に対して開始されるまでシステムに保管されたままです。

書き出しプログラムは、出力待ち行列からスプール・ファイルを一度に 1 つずつ優先順位に基づいて取り出します。出力待ち行列上のスプール・ファイル項目の状況が使用可能 (RDY) の場合にのみ、書き出しプログラムはスプール・ファイルを処理します。出力待ち行列処理 (WRKOUTQ) コマンドを使用して、特定のスプール・ファイルの状況を表示できます。

スプール・ファイルの状況が使用可能の場合、書き出しプログラムは出力待ち行列から項目を取り出して、指定のジョブ区切りまたはファイル区切り (あるいはその両方) とファイル内の出力データを印刷します。スプール・ファイルの状況が使用可能でない場合は、書き出しプログラムはその項目を出力待ち行列に残し、次の項目に進みます。ほとんどの場合、書き出しプログラムは、状況が使用可能のすべてのファイルが出力待ち行列から取り出されるまで、スプール・ファイルの処理を継続します (ジョブ区切りおよびファイル区切りが先にくる)。

書き出しプログラム開始コマンドの AUTOEND パラメーターは、書き出しプログラムが新しいスプール・ファイルが書き込み可能になるのを待つのか、1 つのファイルの処理後に終了するのか、状況が使用可能のすべてのスプール・ファイルが出力待ち行列から取り出された後に終了するのかを決定します。

スプール書き出しプログラム・コマンド:

スプール書き出しプログラムを制御するために使用できるコマンドをここで紹介します。

- ディスケット書き出しプログラム開始 (STRDKTWTR): 指定されたディスケット装置に対してスプール書き出しプログラムを開始して、その装置上のスプール・ファイルを処理します。
- 印刷装置書出プログラム開始 (STRPRTWTR): 指定された印刷装置に対してスプール書き出しプログラムを開始して、その装置上のスプール・ファイルを処理します。
- リモート書き出しプログラム開始 (STRRMTWTR): 出力待ち行列からリモート・システムにスプール・ファイルを送るスプール書き出しプログラムを開始します。
- 書き出しプログラム変更 (CHGWTR): 書き出しプログラムの一部の属性 (用紙タイプ、ファイル区切りページの数、出力待ち行列属性) を変更します。
- 書き出しプログラム保留 (HLDWTR): 書き出しプログラムをレコードの終わり、スプール・ファイルの終わり、またはページの終わりで停止します。
- 書き出しプログラム解放 (RLSWTR): さらに処理を行えるよう、保留されている書き出しプログラムを解放します。
- 書き出しプログラム終了 (ENDWTR): スプール書き出しプログラムを終了し、関連した出力装置をサーバーが使用できるようにします。

注: 一部の機能を定義して、スプール・サポートをさらに提供できます。これらの機能のコマンド、ファイル、およびプログラムのソースおよび資料は、ライブラリー QUSRTOOL (i5/OS の一部としてオプションでインストールされる) に含まれています。

関連情報

印刷装置書出プログラム開始 (STRPRTWTR) コマンド

リモート書き出しプログラムの開始 (STRRMTWTR) コマンド

書き出しプログラム変更 (CHGWTR) コマンド

書き出しプログラム保留 (HLDWTR) コマンド

書き出しプログラム解放 (RLSWTR) コマンド

書き出しプログラム終了 (ENDWTR) コマンド

入力スプーリング:

入力スプーリングは、情報を入力装置から取り、スケジューリングのためにジョブを準備し、項目をジョブ待ち行列に入れます。入力スプーリングを使用すると、通常はジョブの実行時間が短縮され、順次に実行されるジョブ数が増加し、装置のスループットが向上します。

入力スプーリングの主な要素は、次のとおりです。

- **ジョブ待ち行列**: 実行に備えてシステムに投入されるバッチ・ジョブの番号付きリストで、バッチ・ジョブはそこから選択されて実行されます。
- **読み取りプログラム**: ジョブを入力装置またはデータベース・ファイルから取り出し、ジョブ待ち行列に入れる機能です。

バッチ・ジョブが読み取りプログラムによって入力ソースから読み取られると、入力ストリームの中のコマンドはジョブに対する要求としてシステムの中に保管され、インライン・データはインライン・データ・ファイルとしてスプールされ、ジョブに関する項目はジョブ待ち行列に入れられます。 ジョブ情報は、サブシステムによる処理に備えてジョブ待ち行列からジョブ入力が選択されるまで、読み取りプログラムによってシステム内に保管されたままになります。

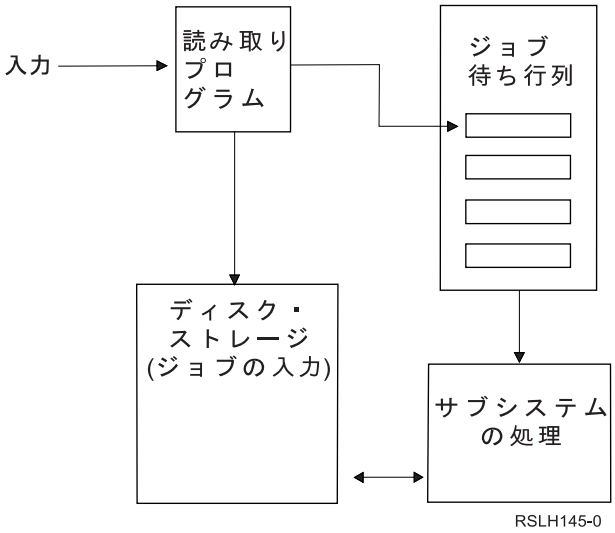


図 2. 入力スプーリング要素の関係

読み取りプログラム機能を使用すれば、ディスケット・ファイルまたはデータベース・ファイルから入力ストリームを読み取ることができます。

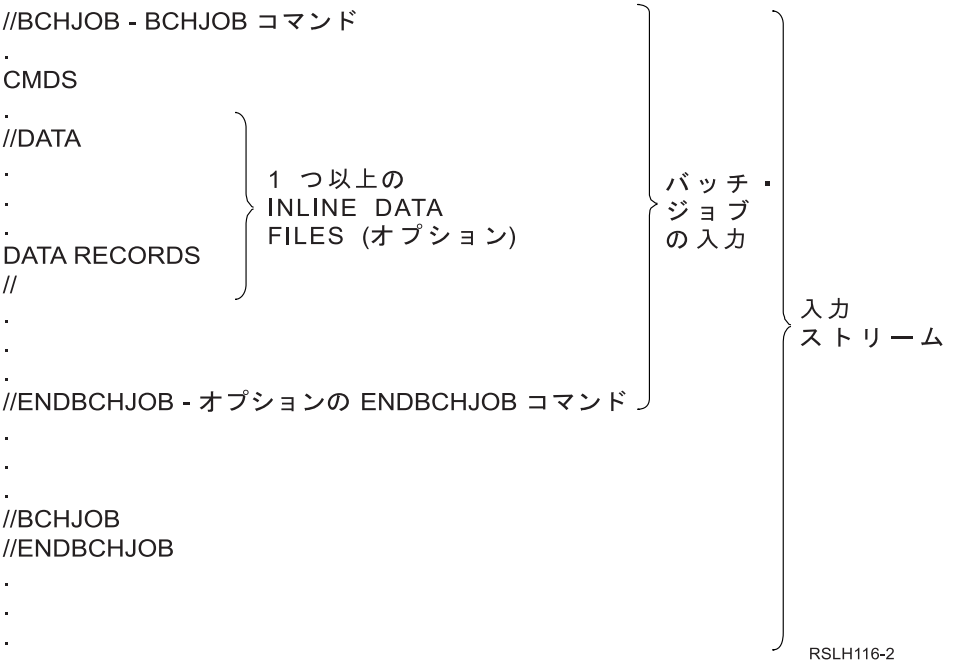


図 3. 入力ストリームの典型的な編成

ジョブが入れられるジョブ待ち行列は、バッチ・ジョブ BCHJOB コマンド、データベース読み取りプログラム開始 STRDBRDR コマンド、またはジョブ記述の中の JOBQ パラメーターで指定されます。
BCHJOB コマンドの JOBQ パラメーターの値は次のとおりです。

- *RDR: ジョブ待ち行列は、STRDBRDR コマンドの JOBQ パラメーターから選択されます。
- *JOBD: ジョブ待ち行列は、ジョブ記述の中の JOBQ パラメーターから選択されます。
- 特定のジョブ待ち行列: 指定された待ち行列が使用されます。

入力ストリームが小さいジョブの場合は、入力スプーリングを使用しないことによって、システム・パフォーマンスの向上を図ることができる場合もあります。 ジョブ投入 (SBMJOB) コマンドで入力ストリームを読み取り、該当するサブシステムの中のジョブ待ち行列にジョブを入れ、スプーリング・サブシステムおよび読み取りプログラムの操作をバイパスします。

ジョブが大きい入力ストリームの読み取りを必要とする場合は、入力スプーリング (ディスケット読み取りプログラム開始 STRDKTRDR または STRDBRDR コマンド) を使用して、ジョブが実際に処理されるのとは別に独立してジョブをインポートできるようにすべきです。

ジョブ入力コマンド:

これらのコマンドを使用して、ジョブをシステムに投入することができます。読み取りプログラム開始コマンドは、ジョブ入力のスプーリングに使用することができます。ジョブ投入コマンドでは、スプーリングは用いません。

- バッチ・ジョブ (BCHJOB): バッチ入力ストリーム内のジョブの開始をマークし、ジョブの操作特性を定義します。
- データ (DATA): インライン・データ・ファイルの開始をマークをします。
- バッチ・ジョブ終了 (ENDBCHJOB): バッチ入力ストリーム内のジョブの終了をマークします。
- 入力終了 (ENDINP): バッチ入力ストリームの終了をマークします。
- データベース・ジョブ投入 (SBMDBJOB): 入力ストリームをデータベース・ファイルから読み取り、その入力ストリーム内のジョブを該当するジョブ待ち行列に入れます。
- ディスケット・ジョブ投入 (SBMDKTJOB): ディスケットから入力ストリームを読み取り、その入力ストリーム内のジョブを該当するジョブ待ち行列に入れます。
- データベース読み取りプログラム開始 (STRDBRDR): 読み取りプログラムを開始して、データベース・ファイルから入力ストリームを読み取り、その入力ストリーム内のジョブを該当するジョブ待ち行列に入れます。
- ディスケット読み取りプログラム開始 (STRDKTRDR) : 読み取りプログラムを開始して、ディスケットから入力ストリームを読み取り、その入力ストリーム内のジョブを該当するジョブ待ち行列に入れます。

  関連情報

  CL コマンド・ファインダー

  バッチ・ジョブ (BCHJOB) コマンド

  データ (DATA) コマンド

  バッチ・ジョブ終了 (ENDBCHJOB) コマンド

  入力終了 (ENDINP) コマンド

  データベース・ジョブ投入 (SBMDBJOB) コマンド

  データベース読み取りプログラム開始 (STRDBRDR) コマンド

インライン・データ・ファイル:

インライン・データ・ファイルは、読み取りプログラムまたはジョブ投入依頼コマンドによってジョブが読み取られるとき、バッチ・ジョブの一部として含まれるデータ・ファイルです。SBMDBJOB または STRDBRDR を使用して、 CL バッチ・ストリーム (実行する CL コマンドのストリーム) に待ち行列を作成します。その CL バッチ・ストリームには、インライン・データ・ファイル (一時ファイル) に配置するデータを含めることができます。ジョブが終了すると、インライン・データ・ファイルは削除されます。

インライン・データ・ファイルは、ジョブの中で、ファイルの開始を //DATA コマンドにより、ファイルの終了をデータ終了区切り文字によって、それぞれ区切られます。

データ終了区切り文字は、ユーザー定義の文字ストリングでもデフォルトの // でもかまいません。 // は 1 桁目および 2 桁目にも存在しなければなりません。データの 1 桁目および 2 桁目に // を含んでいる場合は、// *** END OF DATA のような固有の文字のセットを使用しなければなりません。これを固有のデータ終了区切り文字として指定するには、//DATA コマンドの ENDCHAR パラメーターを次のようにコーディングしなければなりません。

```
ENDCHAR('// *** END OF DATA')
```

注: インライン・データ・ファイルにアクセスすることができるのは、バッチ・ジョブの最初の経路指定ステップ中だけです。バッチ・ジョブにジョブ転送 (TFRJOB) コマンド、ジョブ経路再指定 (RRTJOB) コマンド、またはバッチ・ジョブ転送 (TFRBCHJOB) コマンドが入っている場合は、新しい経路指定ステップでインライン・データ・ファイルにアクセスすることはできません。

インライン・データ・ファイルには、名前が付いていてもいなくても問題ありません。名前のないインライン・データ・ファイルの場合は、QINLINE が //DATA コマンドの中でファイル名として指定されるか、名前は指定されないかどちらかです。名前付きインライン・データ・ファイルの場合、ファイル名が指定されます。

名前付きインライン・データ・ファイル には以下の特徴があります。

- ジョブの中で固有の名前を持ちます。他のインライン・データ・ファイルが同じ名前を持つことはありません。
- ジョブの中で複数回使用できます。
- オープンのたびに最初のレコードに位置合わせされます。

名前付きインライン・データ・ファイルを使用するには、プログラムでファイル名を指定するか、プログラムで指定されているファイル名を指定変更コマンドを使用してインライン・データ・ファイルの名前に変更する必要があります。ファイルは、入力専用としてオープンされます。

名前のないインライン・データ・ファイル には以下の特性があります。

- 名前は QINLINE です。(1 つのバッチ・ジョブの中では、名前のないインライン・データ・ファイルには、すべて同一名が与えられます。)
- ジョブの中で使用できるのは 1 回だけです。
- 1 つのジョブの中に名前のないインライン・データ・ファイルが複数個含まれるときは、それらのファイルはファイルのオープン時と同じ順序で入力ストリームの中になければなりません。

名前のないインライン・データ・ファイルを使用するには、次のいずれかのようにします。

- プログラムで QINLINE を指定します。
- 指定変更ファイル・コマンドを使用して、プログラムの中で指定されているファイル名を QINLINE に変更します。

使用している高水準言語が 1 つのプログラム内で固有のファイル名を必要とする場合は、QINLINE をファイル名として使用できるのは 1 回だけです。名前のないインライン・データ・ファイルを複数個使用する必要がある場合は、指定変更ファイル・コマンドをプログラムの中で使用して、その他の名前のないインライン・データ・ファイルに対して QINLINE を指定することができます。

注: コマンドを条件付きで実行し、名前のないインライン・データ・ファイルを複数処理する場合、名前のない間違ったインライン・データ・ファイルを使用すると、結果は予測できません。

関連概念

『インライン・データ・ファイルを開く際の考慮事項』
インライン・データ・ファイルを開くときはこれらの要素を検討する必要があります。

インライン・データ・ファイルを開く際の考慮事項:

インライン・データ・ファイルを開くときはこれらの要素を検討する必要があります。

- レコード長は、入力レコードの長さを指定します。 (レコード長はオプションです。) レコード長がデータの長さを超える場合、メッセージがプログラムに送信されます。データにはブランクが埋め込まれます。レコード長がデータ長より短い場合、レコードは切り捨てられます。
- ファイルがプログラムで指定されると、システムはそのファイルを、ライブラリー内から検索する前に、名前付きインライン・データ・ファイルとして検索します。 したがって、名前付きインライン・データ・ファイルが、インライン・データ・ファイルではないファイルと同じ名前を持っている場合、ファイル名がライブラリー名で修飾されていないとしても、必ずそのインライン・データ・ファイルが使用されます。
- 名前付きインライン・データ・ファイルは、ファイル作成コマンドまたはファイル指定変更コマンドで SHARE(*YES) を指定すると、同じジョブ内のプログラム間で共有できます。たとえば、INPUT という名前のファイルと SHARE(*YES) を指定するファイル指定変更コマンドが、INPUT という名前のインライン・データ・ファイルを持つバッチ・ジョブ内にある場合、ファイル名 INPUT を指定している、そのジョブ内で実行しているプログラムは、その同じ名前のインライン・データ・ファイルを共有します。名前がないインライン・データ・ファイルを同じジョブ内のプログラム間で共有することはできません。
- インライン・データ・ファイルを使用する場合、//DATA コマンドに正しいファイル・タイプを指定していることを必ず確認してください。 たとえば、ファイルがソース・ファイルとして使用される場合、//DATA コマンドに指定するファイル・タイプはソースでなければなりません。
- インライン・データ・ファイルは入力に対してのみ開く必要があります。

関連タスク

82 ページの『インライン・データ・ファイル』
インライン・データ・ファイルは、読み取りプログラムまたはジョブ投入依頼コマンドによってジョブが読み取られるとき、バッチ・ジョブの一部として含まれるデータ・ファイルです。SBMDBJOB または STRDBRDR を使用して、 CL バッチ・ストリーム (実行する CL コマンドのストリーム) に待ち行列を作成します。その CL バッチ・ストリームには、インライン・データ・ファイル (一時ファイル) に配置するデータを含めることができます。ジョブが終了すると、インライン・データ・ファイルは削除されます。

ジョブ・ログ

ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

保留形式の場合、完了ジョブのジョブ・ログは、他のジョブ (サブシステム、システム操作など) がこの完了ジョブと対話するにつれて変更されることがあります。スプール形式の場合には、ジョブ・ログは (ある特定の瞬間の) スナップショットであり、変更することはありません (ジョブ・ログ表示 (DSPJOBLOG) コマンドによって作成されるスプール・ファイル、またはジョブがアクティビティーを完了した後に作成されるスプール・ファイルなど)。

各ジョブには関連するジョブ・ログがあり、ジョブに関する以下の情報を含めることができます。

- ジョブ内のコマンド

- CL プログラム内のコマンド (CL プログラムが LOG(*YES) オプションまたは LOG(*JOB) オプションで作成され、ジョブ変更 (CHGJOB) コマンドが LOGCLPGM(*YES) オプションで実行された場合)
- 要求元に送信され、プログラム・メッセージ待ち行列からは除去されないすべてのメッセージ (メッセージおよびメッセージのヘルプ・テキスト)

ジョブの終了時、ジョブ・ログはスプール・ファイル QPJOBLOG に書き込まれ、印刷可能です。しかし、ジョブ・ログを生成するということは、それを印刷する、またはスプール・ファイルを作成することを必ずしも意味しません。(たとえば、ジョブ・ログ制御 QMHCTLJL API は、ジョブの終了時に出力ファイルに書き込まれるジョブ・ログを指定するのに使用できます。)

生成されるジョブ・ログ数を減らし、さらにリソースの競合 (出力待ち行列など) を少なくできます。このようにすると、ジョブ・ログの生成によって生じるリソースの使用量を減らせます。

関連概念

218 ページの『ジョブ・ログの管理』
システム上のほとんどのジョブには、それに関連したジョブ・ログがあります。 ジョブ・ログには、いつジョブが開始したか、いつジョブが終了したか、どのコマンドが実行中か、障害通知およびエラー・メッセージなど、さまざまなことが記録されています。この情報を見ると、ジョブ・サイクルがどのように行われているかがよく分かります。

218 ページの『ジョブ・ログ・サーバーの管理』
QSYSWRK サブシステムは、ジョブ・ログ・サーバーを制御します。また幾つかのタスクを実行すると、ジョブ・ログ・サーバーをカスタマイズまたは管理することができます。

関連タスク

228 ページの『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 → 「**プリンター出力**」の下にあります。

227 ページの『バッチ・ジョブのログ情報の制御』
バッチ・アプリケーションの場合に、ログに記録される情報量を変更できます。 IBM 提供のサブシステム QBATCH のジョブ記述で指定されるログ・レベル (LOG(40 *NOLIST)) は、ジョブが異常終了した場合に完全なログを提供します。ジョブが正常に完了した場合、ジョブ・ログは生成されません。

226 ページの『ジョブのログ・レベルの変更』
ジョブのログ・レベルは、ログに記録されるメッセージ・タイプの特定の組み合わせに割り当てられる数値レベルです。ジョブ記述内のログ・レベルは、文字ベース・インターフェースを使用して変更できます。ただし、特定のジョブのログ・レベルを変更したい場合は、System i ナビゲーター 内の「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」 ウィンドウを使用します。

関連情報

経験報告: スプール・パフォーマンスの考慮点

ジョブ・ログの作成方法

ジョブ・ログは必要なときに使用できますが、ジョブ・ログの必要がない場合はそれを生成する作業は実行されません。

LOG パラメーターには、メッセージ (またはロギング) レベル、メッセージ重大度、およびメッセージ・テキストのレベルという、3 つの要素があります。 これらの各要素には、結合された場合に、ジョブによってジョブ・ログに送信される情報の量およびタイプを決定する固有の値があります。

たとえば、テスト要素の *NOLIST 値によって、ジョブが正常に終了した場合はジョブ・ログは生成されません。 (ジョブ・ログは保留状態になりません。) ジョブが異常終了した場合 (ジョブ終了コードが 20 以上の場合)、ジョブ・ログが生成されます。ジョブ・ログに出力されるメッセージには、メッセージ・テキストおよびメッセージ・ヘルプの両方が含まれます。

ジョブ・ログを生成するものを制御できます。これは LOGOUTPUT パラメーターを指定して実行できます。 ジョブが完了した場合、3 つのアクションの 1 つが実行され、ジョブ・ログの作成方法が影響を受けます。以下は、LOGOUTPUT パラメーターの値です。

- **ジョブ・ログ・サーバーがジョブ・ログを生成します:** (*JOBLOGSVR)
- **ジョブ自体がジョブ・ログを生成します:** ジョブがその固有のジョブ・ログを生成できない場合、ジョブ・ログはジョブ・ログ・サーバーによって生成されます。 (*JOBEND)
- **ジョブ・ログは生成されません:** ジョブ・ログは除去されるまで保留状態になります。(*PND)

注: これらの値は、ジョブ・メッセージ待ち行列フルのアクションに *PRTWRAP が指定されている場合に、メッセージ待ち行列が満杯状態になった時点で生成されるジョブ・ログには影響を与えません。ジョブ・メッセージ待ち行列内のメッセージはスプール・ファイルに書き込まれ、そのスプール・ファイルからジョブ・ログを印刷できます。ただしこれは、ジョブ・ログ出力制御 (QMHCTLJL) API がジョブ内で使用され、ジョブ・ログ内のメッセージがデータベース・ファイルに書き込まれるように指定されていない場合に限ります。

何によってジョブ・ログ・パラメーターを制御するか

ジョブの開始時に、ジョブはその LOGOUTPUT 値をジョブ記述から入手します。 ジョブ記述が *SYSVAL (CRTJOBD のデフォルト) を指定している場合、ジョブは、ジョブ・ログ出力 (QLOGOUTPUT) システム値で指定されているジョブ・ログ出力値を使用します。 (ジョブ・ログ出力 (QLOGOUTPUT) システム値の出荷時の値は *JOBEND ですが、推奨値は *JOBLOGSVR です。) ジョブがその LOGOUTPUT ジョブ属性を設定した後に、ジョブ記述またはシステム値に加えられた変更は、アクティブ・ジョブに影響を与えません。 システム値またはジョブ記述に加えられた変更は、変更後にシステムに投入されたジョブに対して有効です。

ジョブ変更 (CHGJOB) コマンドまたは API (QWTCHGJB) を使用して、LOGOUTPUT ジョブ属性を、それがジョブ内に設定された後に変更できます。 ジョブに加えた変更は即時に有効になります。

選択した方法に関係なく、ジョブ・ログを処理するオプションは同じです。ジョブには、ジョブ・ログを生成しない (*PND)、ジョブにジョブ・ログを生成させる (*JOBEND)、またはジョブ・ログ・サーバーにジョブ・ログを生成させる (*JOBLOGSVR) が設定可能です。

関連タスク

224 ページの『特定のジョブ・ログの作成の停止』
特定のジョブ・ログの作成のみを停止する場合は、ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドを使用しないでください。 ENDLOGSVR コマンドは、すべてのジョブ・ログ・サーバーを終了するため、すべてのジョブ・ログの作成が停止します。

225 ページの『ジョブ・ログの作成の抑制』
ジョブ・ログの作成の抑制は、ジョブ・ログが不要であることが分かっていて、システム・リソースを節約する場合に役立ちます。ジョブ・ログを作成しないことを指定すると、ジョブ・ログは生成されなくなり、保留ジョブ・ログ除去 (QWTRMVJL) コマンドまたはジョブ終了 (ENDJOB) コマンドで除去しない限り、保留のままになります。

225 ページの『ジョブ・ログ内の情報の制御』
問題を処理するときに、頻繁に問題が生じるジョブについては記録される情報量を最大にしたい場合が

あります。別の状況として、正常に完了したジョブについてはジョブ・ログを作成したくない場合もあります。あるいは、通知メッセージは除外したいという場合もあります。

ジョブ・ログ保留

ジョブ・ログ保留状態は、これまで何年も使用されてきました。 ジョブのジョブ・ログ属性が *PND の場合、ジョブ・ログは生成されません。特定のジョブのジョブ・ログをどのように、またどのような状況下で生成するかを制御することができます。

この機能は、システムが制限状態に置かれている場合に役立ちます。システムが制限状態になると、サブシステムは終了し、潜在的な数千ものジョブが同時に終了する可能性があります。これにより出力リソースには大きな負荷がかかる可能性があります。これらのジョブ・ログの生成を避けることで、それらのリソースへの影響を大幅に削減できます。

この機能を使用できる別の例として、通信障害時があります。 同じジョブ・ログ・エラー・メッセージを生成する多くの類似のジョブが存在する場合があります。すべてのジョブについてジョブ・ログがスプール・ファイルを生成しないように設定することができます。 万一通信障害がある場合には、「ジョブ・ログ処理 (WRKJOBLOG)」コマンドを使用して、印刷するログを決定できます。 さらに、「ジョブ・ログ処理 (WRKJOBLOG)」画面を使用して、ジョブ・ログを管理できます。

ジョブは、「システム電源遮断 (PWRDWNSYS)」コマンドの作動により、ジョブ・ログ保留状態になっている場合があります。 System i ナビゲーター のユーザー・インターフェースは、これらのジョブについて、状況を「Completed - Job log pending (完了 - ジョブ・ログ保留)」と表示します。 これは、文字ベースのインターフェースの状況 *OUTQ のサブセットです。

これらの拡張を活用することで、生成されるジョブ・ログの数を減らし、それに伴いリソースの競合を減らすことができます。この結果としてシステム・パフォーマンスを向上させることができます。

関連概念

51 ページの『同時に複数のジョブが終了する場合』
複数のジョブが同時に終了することが時折あります。たとえば、ネットワーク・エラーが生じて、ジョブ属性が *ENDJOB または *ENDJOBNOLIST に設定されます。ジョブの終了に加えて、以下の装置リカバリー・アクションが生じます。

関連タスク

229 ページの『保留ジョブ・ログのクリーンアップ』
保留ジョブ・ログからジョブをクリーンアップまたは除去するにはいくつかの方法があります。ジョブは、最大ログ項目 (LOGLMT) パラメーターの値を 0 に設定して終了できます。 ジョブがすでに終了している場合、保留ジョブ・ログ除去 (QWTRMVJL) API を実行できます。ジョブ・ログ処理 (WRKJOBLOG) コマンドも使用できます。

229 ページの『ジョブ・ログ保留からのプリンター出力の生成』
System i ナビゲーター 「**Job Properties - Job Log (ジョブ・プロパティー - ジョブ・ログ)**」設定の「**Produce a job log (ジョブ・ログの生成)**」フィールドが選択されていないジョブでは、ジョブ・ログは生成されません。 その代わり、ジョブ・ログはジョブ・ログ保留状態になります。ジョブ・ログ保留状態にあるジョブ・ログからプリンター出力を生成するには、文字ベース・インターフェースを使用します。

ジョブ・ログ・サーバー

通常、ジョブ・ログ・サーバーはジョブのジョブ・ログをスプール・ファイルに書き込みます。ジョブ・ログを印刷装置または出力ファイルに経路指定できますが (QMHCTLJL、ジョブ・ログ制御 API を使用してそのように指定する場合)、ジョブ・ログを生成するにはこの方法は推奨されていません。

System i ナビゲーターの「**実行管理機能**」 → 「**Server Jobs (サーバー・ジョブ)**」画面、または「**実行管理機能**」 → 「**アクティブ・ジョブ**」画面から、ジョブ・ログ・サーバーに関する情報を表示できます。(ジョブ・ログ・サーバーで実行されているジョブを簡単に識別するには、画面に「サーバー」列が含まれるようにしてください。)

一度にアクティブにできるジョブ・ログ・サーバーの最大数は 30 です。システム内の他のサーバーと同様の仕方で、追加のジョブ・ログ・サーバーを開始して管理できます。文字ベース・インターフェース・コマンド STRLOGSVR を使用するとこれが可能です。

ジョブ・ログ・サーバーの開始方法

デフォルトでは、ジョブ・ログ・サーバーは QSYSWRK サブシステムの開始時に自動的に開始されます。QSYSWRK サブシステムの終了時には必ずサーバーは終了します。

ジョブ・ログ・サーバー開始 (STRLOGSVR) コマンドは、ジョブ・ログ・サーバーを開始します。 ジョブ・ログ・サーバーは、ジョブ・ログ保留状態にあるものの、属性 *PND がないジョブのジョブ・ログを書き込みます。ジョブ・ログ・サーバーは、スプール・ファイル、印刷装置、または出力ファイルのいずれかに、ジョブのジョブ・ログを書き込みます (QMHCTLJL、ジョブ・ログ制御 API を使用してそのように指定する場合)。

関連タスク

219 ページの『ジョブ・ログ・サーバーの再構成』
出荷時には、ジョブ・ログ・サーバーは QSYSWRK で実行します。 QSYSWRK は絶えずアクティブです。パフォーマンスを強化するには、別のサブシステムでジョブ・ログ・サーバーが実行するように再構成することもできます。

220 ページの『ジョブ・ログ・サーバーの開始』
デフォルトで、ジョブ・ログ・サーバーは、QSYSWRK サブシステムが始動すると自動的に開始します。ジョブ・ログ・サーバー開始 (STRLOGSVR) コマンドを使用して、ジョブ・ログ・サーバーを手動で開始できます。

219 ページの『ジョブ・ログ・サーバーの終了』
ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドは、ジョブ・ログ・サーバーを終了するために使用します。 ジョブ・ログ・サーバーは、ジョブ・ログ保留状態にあるジョブのジョブ・ログを書き込みます。このコマンドの発行時に、複数のジョブ・ログ・サーバーのジョブがアクティブの場合、すべてのジョブ・ログ・サーバーのジョブが終了します。

関連情報

Control Job Log Output (QMHCTLJL) API

ジョブ・ログの表示特性

System i ナビゲーター には、ジョブ・ログおよびジョブ・ログのメッセージを表示できる、扱いやすく読み取りやすいユーザー・インターフェースが備えられています。 さらに、文字ベース・インターフェースを使用してもジョブ・ログを表示できます。

「Job Log - Columns (ジョブ・ログ - 列)」ウィンドウを使用すると、ジョブ・ログ・リスト内に表示される列を制御できます。(「**実行管理機能**」 → 「**アクティブ・ジョブ**」 → **ジョブを右マウス・ボタンでクリックして「ジョブ・ログ」を選択する** → 「**View (表示)**」**メニュー** → 「**Customize this view (この表示をカスタマイズ)**」 → 「**Columns (列)**」) ジョブ・ログ・リストで表示するために選択できる列は、以下のとおりです。

メッセージ ID	From Program (プログラムから)

Message (メッセージ)	Request Level (要求レベル)
Sent (送信済み)	Severity (重大度)
Thread (スレッド)	To Program (プログラムへ)
タイプ	

文字ベース・インターフェース

ジョブ・ログ表示 (DSPJOBLOG) コマンドを使用すると、「ジョブ・ログ」画面が表示されます。この画面には、プログラム名が以下のような特殊記号を伴って表示されます。

>>	実行中のコマンドまたは次に実行するコマンド。たとえば、CL または高水準言語プログラムが呼び出されると、そのプログラムへの呼び出しが表示されます。
>	コマンドは処理を完了しました。
. .	コマンドは、まだ処理されていません。
?	返信メッセージ。この記号は、返信を必要とするメッセージおよび応答されたメッセージにマークされます。

ジョブ・ログ・ヘッディング:

ジョブ・ログ・ヘッディングは、印刷されるジョブ・ログの各ページの上部に配置されます。こうしたヘッディングには、ジョブ・ログが適用されるジョブ、および各項目の特性が示されます。以下は、ジョブ・ログ・ヘッディング内に示すことが可能な項目のリストです。

- ジョブの完全修飾名 (ジョブ名、ユーザー名、およびジョブ番号)
- ジョブを開始するのに使用するジョブ記述名
- ジョブが開始された日時
- メッセージ ID
- メッセージ・タイプ
- メッセージ重大度
- 各メッセージが送信された日時
- メッセージ。第 2 レベル・テキストが組み込まれるようにロギング・レベルで指定する場合、第 2 レベル・テキストはメッセージの下の後続行に表示されます。
- メッセージまたは要求の送信元プログラム
- マシン・インターフェースの命令番号またはメッセージの送信先プログラムに対するオフセット

  注: マシン・インターフェースの命令番号は、エスケープ、通知、および診断の各メッセージでのみ表示されます。他のすべてのメッセージ・タイプの場合、マシン・インターフェースの命令番号はゼロに設定されます。
- ジョブが APPC を使用する場合、ヘッディングには APPC の作業単位 ID を示す行が含まれます。

メッセージ:

メッセージには、ジョブ名、メッセージ・タイプ、送信された日時、発生したアクション、および問題を修正するのに必要なアクションが含まれています。これは、サーバーで生じる問題をトラブルシューティングしようとする際に役立ちます。System i ナビゲーター によって、サーバー・ジョブのジョブ・ログにアクセスするすることができます。 メッセージは、警告メッセージとジョブ・ログに記録されたメッセージという 2 つのカテゴリーに分類されます。

警告メッセージ - このメッセージは、即時アクションが必要なので QSYSOPR に送信されます。このメッセージには、問題、原因、および必要な回復アクションが含まれています。たとえば、サーバーが開始に失敗した場合や不意に終了する場合などです。一部のサーバーは、警告メッセージを QSYSOPR に送信します。こうしたメッセージには、メッセージ記述で定義される警報オプション (ALROPT) が存在します。警報を使用すると、警告メッセージを集中的に処理できます。

ジョブ・ログに記録されたメッセージ - このメッセージは診断的な性格のものなので、重大ではありませんが、ユーザーが実行した何らかのアクションに対する警告です。このメッセージは、システムによって生成されることもあれば、ユーザーが作成した場合もあります。

メッセージ・ロギング・レベル

メッセージ・ロギング・レベルは、ジョブ用にログに記録すべきメッセージおよびメッセージ・タイプを判別します。以下の表は、各レベルが意味する事柄について説明しています。

レベル	説明
レベル 1	メッセージ重大度値以上の重大度を有するジョブの外部メッセージ待ち行列に送信されるすべてのメッセージ。 (System i ナビゲーター では、メッセージ重大度 (0-99) 値は「Job Properties - Job Log (ジョブ・プロパティー - ジョブ・ログ)」ウィンドウに表示されます。 これは、制御可能な値です。)
レベル 2	レベル 1 の資格を満たすすべてのメッセージ、および結果としてメッセージ重大度値以上の高水準メッセージになる要求メッセージ。 **注:** 高水準メッセージとは、要求メッセージを受け取るプログラムのプログラム・メッセージ待ち行列に送信されるメッセージです。(たとえば、QCMD は要求メッセージを受け取る IBM 提供の要求処理プログラムです。)
レベル 3	レベル 1 またはレベル 2 の資格を満たすすべてのメッセージ、およびすべての要求メッセージ。さらに、(「Job Properties - Job Log (ジョブ・プロパティー - ジョブ・ログ)」ウィンドウの)「**Log commands from CL programs box (CL プログラムからのログ・コマンド・ボックス)**」がチェックされている場合、CL プログラムからの任意のコマンドも含まれます。 **注:** 「**Log commands from CL programs box (CL プログラムからのログ・コマンド・ボックス)**」は、CL プログラムのログ属性に相当します。
レベル 4	すべての要求メッセージ、およびトレース・メッセージを含む、メッセージ・ロギング重大度以上の重大度を持つすべてのメッセージ。さらに、(「Job Properties - Job log (ジョブ・プロパティー - ジョブ・ログ)」ウィンドウの)「**Log commands from CL programs box (CL プログラムからのログ・コマンド・ボックス)**」がチェックされている場合、CL プログラムからの任意のコマンドも含まれます。 **注:** 「**Log commands from CL programs box (CL プログラムからのログ・コマンド・ボックス)**」は、CL プログラムのログ属性に相当します。

関連タスク

226 ページの『ジョブのログ・レベルの変更』
ジョブのログ・レベルは、ログに記録されるメッセージ・タイプの特定の組み合わせに割り当てられる数値レベルです。ジョブ記述内のログ・レベルは、文字ベース・インターフェースを使用して変更できます。ただし、特定のジョブのログ・レベルを変更したい場合は、System i ナビゲーター 内の「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」 ウィンドウを使用します。

対話式ジョブのログ

IBM 提供のジョブ記述 QCTL、QINTER、および QPGMR すべてには、LOG(4 0 *NOLIST) のログ・レベルがありますので、すべてのメッセージ・ヘルプ・テキストがジョブ・ログに書き込まれます。しかしジョブが正常に終了する場合には、ジョブ・ログは印刷されません (SIGNOFF コマンドで *LIST を指定する場合を除く)。

ディスプレイ装置ユーザーが IBM 提供のメニューまたはコマンド入力画面を使用すると、すべてのエラー・メッセージが表示されます。ディスプレイ装置ユーザーがユーザー作成の初期プログラムを使用すると、モニターされていないメッセージによって初期プログラムが終了し、ジョブ・ログが生成されます。しかし初期プログラムがメッセージをモニターしている場合には、そのメッセージを受信する際に制御を受け取ります。この場合、ジョブ・ログが生成されて生じた特定のエラーを判別できるようにするのは重要なことです。

たとえば、初期プログラムで、デフォルトが *NOLIST のサインオフ・オプションが含まれるメニューが表示されると想定します。初期プログラムはすべての例外をモニターし、例外が生じる場合にサインオフ・オプションを *LIST に変更する変数変更 (CHGVAR) コマンドを組み込みます。

```
PGM
DCLF MENU
DCL &SIGNOFFDPT TYPE(*CHAR) LEN(7)
VALUE(*NOLIST)
.
.
.
MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
PROMPT: SNDRCVF RCDFMT(PROMPT)
CHGVAR &IN41 '0'
.
.
.
IF (&OPTION *EQ '90') SIGNOFF
LOG(&SIGNOFFOPT);
.
.
.
GOTO PROMPT
ERROR: CHGVAR&SIGNOFFOPT '*LIST'
CHGVAR &IN41 '1'
GOTO PROMPT
ENDPGM
```

例外が発生すると、CHGVAR コマンドは SIGNOFF コマンドのオプションを *LIST に変更し、標識を使用します。 この標識は、予期しないエラーが発生したことを示し、ディスプレイ装置ユーザーが行うべきことを指示するメッセージを表示する、定数の条件を定めるのに使用できます。

QHST ヒストリー・ログ

ヒストリー (QHST) ログは、メッセージ待ち行列およびログ・バージョンと呼ばれる物理ファイルで構成されます。ログ・メッセージ待ち行列に送信されるメッセージは、システムによって現行ログ・バージョン物理ファイルに書き込まれます。

ヒストリー・ログ (QHST) には、システム、サブシステム、ジョブ情報、装置状況、およびシステム・オペレーター・メッセージなどのシステム・アクティビティーのトレースの概要が含まれています。メッセージ待ち行列は QHST です。

ログ・バージョン

各ログ・バージョンは、以下のような名前の物理ファイルです。

```
Qxxxyydddn
```

ここで、

xxx は、ログ・タイプ (HST) の 3 文字の説明

yyddd は、ログ・バージョンが作成されたユリウス日付

n は、ユリウス日付の順序番号 (0 から 9 または A から Z)

ログ・バージョンがいっぱいになると、ログの新しいバージョンが自動的に作成されます。

注: ヒストリー・ログのログ・バージョン内のレコード数は、ヒストリー・ログ内のレコードの最大数 (QHSTLOGSIZ) システム値で指定します。このシステム値は、新しいバージョンを日ごとに作成する *DAILY オプションもサポートします。

ヒストリー・ログのフォーマット:

データベース・ファイルは、システム・ログに送信されるメッセージを保管するために使用されます。物理ファイル内のすべてのレコードは同じ長さであり、ログに送信されるメッセージは異なる長さなので、メッセージは複数のレコードにまたがる場合があります。

メッセージのそれぞれのレコードには、以下の 3 つのフィールドがあります。

- システム日時 (長さ 8 の文字フィールド)。これは内部フィールドです。変換された日時もメッセージ内に入ります。
- レコード番号 (2 バイト・フィールド)。たとえば、フィールドには、最初のレコードには 16 進 0001、2 番目のレコードには 16 進 002、という具合に番号が入ります。
- データ (長さ 132 の文字フィールド)。

3 番目のフィールドのフォーマット (データ):

表 1. 最初のレコードの3 番目のフィールドのフォーマット

内容	タイプ	長さ	レコード内の位置
ジョブ名	文字	26	11-36
変換日時	文字	13	37-49
メッセージ ID	文字	7	50-56
メッセージ・ファイル名	文字	10	57-66
ライブラリー名	文字	10	67-76
メッセージ・タイプ	文字	2	77-78
重大度コード	文字	2	79-80
プログラム名の送信	文字	12	81-92
プログラム名の受信	文字	10	97-106
プログラム命令番号の受信	文字	4	107-110
メッセージ・テキスト長	バイナリー	2	111-112
メッセージ・データ長	バイナリー	2	113-114
予約済み	文字	28	115-142

表 2. 残りのレコードの 3 番目のフィールドのフォーマット (データ)

内容	タイプ	長さ
メッセージ	文字	変数 (この長さは最初のレコードに指定され (位置 111 および 112)、132 より長くすることはできません。)
メッセージ・データ	文字	変数 (この長さは最初のレコードに指定されます (位置 113 および 114)。)

新規バージョンのログの開始時に、メッセージが分割されることはありません。メッセージの先頭および最終レコードは、常に同じ QHST バージョンです。

QHST ファイル処理

QHST ファイルを処理するために高水準言語プログラムを使用する場合、メッセージ・データは、同じメッセージを使用するごとに変数位置で開始されることに留意してください。この理由は、メッセージには置き換え可能な変数が含まれており、メッセージの実際の長さは変化するからです。

ただし、メッセージ CPF1124 (ジョブ開始) およびメッセージ CPF1165 (ジョブ完了) の場合、メッセージ・データは常に 3 番目のレコードの位置 11 で開始されます。

パフォーマンス情報および QHST:

パフォーマンス情報は、メッセージ CPF1164 にテキストとしては表示されません。このメッセージは QHST ログ内にあるので、このデータを取得するためのアプリケーション・プログラムを作成できます。

パフォーマンス情報は、可変長の置換テキスト値として渡されます。つまり、このデータは最初の項目内の構造内に入れられ、それがデータの長さとなります。この長さフィールドのサイズは、長さには含まれません。

時刻および日付: 構造内の最初の日付フィールドは、ジョブがシステムに投入された時刻と日付、さらにはジョブの最初の経路指定ステップが開始された時刻と日付です。時刻は、hh:mm:ss という形式です。この例での時刻区切り記号は、コロンです。この区切り記号は、日時 (QTIMSEP) システム値で指定された値によって決まります。日付は日時 (QDATFMT) システム値で定義された形式で、区切り記号は日時 (QDATSEP) システム値で定義されます。 この構造では、ジョブがシステムに投入された時刻および日付は、ジョブ開始日および時刻よりも先行します。ジョブがシステムに投入された日時は、ジョブが開始されたことをシステムが認識する時刻です (ジョブ構造は、ジョブのために設定されるものです)。対話式ジョブの場合、ジョブ入力時間はシステムによってパスワードが認識される時刻です。バッチ・ジョブの場合、バッチ・ジョブ (BCHJOB) またはジョブ投入 (SBMJOB) コマンドが処理される時間です。 モニター・ジョブ、読み取りプログラムまたは書き込みプログラムの場合には該当する開始コマンドが処理される時間で、自動開始ジョブの場合はサブシステムが開始される間になります。

応答時間合計とトランザクション数: 時刻と日付の次は、応答時間合計とトランザクション数です。応答時間合計は秒単位で、ワークステーションで ENTER キーを押してから次の画面が表示されるまでの間の、ジョブが処理されたすべての間隔の累算値が含まれます。この情報は、アクティブ・ジョブ処理 (WRKACTJOB) 画面に表示される情報と類似しています。このフィールドは、対話式ジョブでのみ意味を持ちます。

さらに、システム障害やジョブの異常終了の場合には、この合計には最後のトランザクションが含まれない可能性があります。この場合、ジョブ終了コードは 40 以上になります。またトランザクション・カウントは、ジョブの際にシステムがカウントする応答時間間隔の数で、コンソール・ジョブではなく対話式ジョブに対してのみ意味を持ちます。

同期補助入出力操作の数: 同期補助入出力操作の数が、トランザクション数の後に続きます。マルチスレッドのあるジョブでは、この値には初期スレッドの同期補助入出力操作のみが含まれます。これは、以下の違いを除いて、WRKACTJOB 画面に表示される AUXIO フィールドと同じです。

- WRKACTJOB 画面には、現行の経路指定ステップの初期スレッドの値が表示されます。
- QHST メッセージには、ジョブ内の各経路指定ステップの初期スレッドの累積合計が含まれます。

ジョブが終了コード 70 で終了する場合、この値には最後の経路指定ステップのカウントが含まれない場合があります。さらに、(バッチ・ジョブ転送 (TFRBCHJOB) を使用して) ジョブが IPL の間に存在している場合、IPL の後にアクティブ状態になる前にジョブは終了し、値は 0 になります。

スプール・ファイル

スプール・ファイルは、出力データを印刷可能になるまで保持します。スプール・ファイルは、プログラムまたは装置がデータを処理できるようになるまで、装置からデータを収集します。プログラムはスプール・ファイルを、実装置で読み取り/書き込みを実行しているかのように使用します。これは入出力スプーリングです。

入力スプーリングは、データベースまたはディスケット・ファイルに対してシステムが実行します。読み取りプログラムと呼ばれる IBM 提供のプログラムは、スプーリング・サブシステムで開始され、装置からバッチ・ジョブ・ストリームを読み取り、ジョブをジョブ待ち行列上に入れます。

出力スプーリングはプリンターに対して実行されます。印刷装置書出プログラムと呼ばれる IBM 提供のプログラムは、スプーリング・サブシステムで開始され、その出力待ち行列からスプール・ファイルを選択し、スプール出力ファイルの記録をプリンターに書き込みます。

ジョブの終了時に、ジョブ・ログはスプール・ファイル QPJOBLOG に書き込まれ、印刷できるようになります。

ジョブ会計

ジョブ会計機能はデータを収集し、システムの使用者および使用しているシステム・リソースを判別することができます。さらに、システム全体の使用を評価するのにも役立ちます。ジョブ会計は、オプションです。ジョブ会計をセットアップするには、特定のステップを行う必要があります。システムに対して、ジョブ・リソースの会計データ、プリンター・ファイルの会計データ、またはその両方を収集するように要求できます。さらに、ユーザー・プロファイルまたは特定のジョブに対して、会計コードを割り当てることもできます。

一般的なジョブ会計データは、システムで実行中のジョブ、および処理装置、印刷装置、ディスプレイ装置など使用中のリソース、ならびにデータベース機能および通信機能の詳細を示します。

ジョブ会計統計は、システム会計ジャーナル QSYS/QACGJRN に作成されたジャーナル項目を使用して保持されます。ジャーナル・レシーバーの保管、ジャーナル・レシーバーの変更、および古いジャーナル・レシーバーの削除などのジャーナル管理操作の実行方法を把握してください。

ジョブ会計データを分析する場合、ジャーナル表示 (DSPJRN) コマンドを使用して QACGJRN ジャーナルからデータを抽出する必要があります。 このコマンドを使用すると、データベース・ファイルに項目を書

き込むことができます。データを分析するには、アプリケーション・プログラムを作成するか、QUERY ユーティリティーなどのユーティリティーを使用しなければなりません。

関連概念

230 ページの『ジョブ会計の管理』
ジョブ会計機能は、デフォルトではアクティブ状態になっていません。この機能をセットアップするには、最初に幾つかのステップが必要です。以下の情報では、ジョブ会計をセットアップして、ジョブ会計に関連する最も一般的なタスクの幾つかを実行する方法が取り上げられています。

関連情報

ジャーナル管理

ジャーナル処理のセットアップ

ジョブ会計の作動方法

ジョブ会計の作動方法の概説では、3 つの異なるジョブ項目をシステムに投入するものとします。

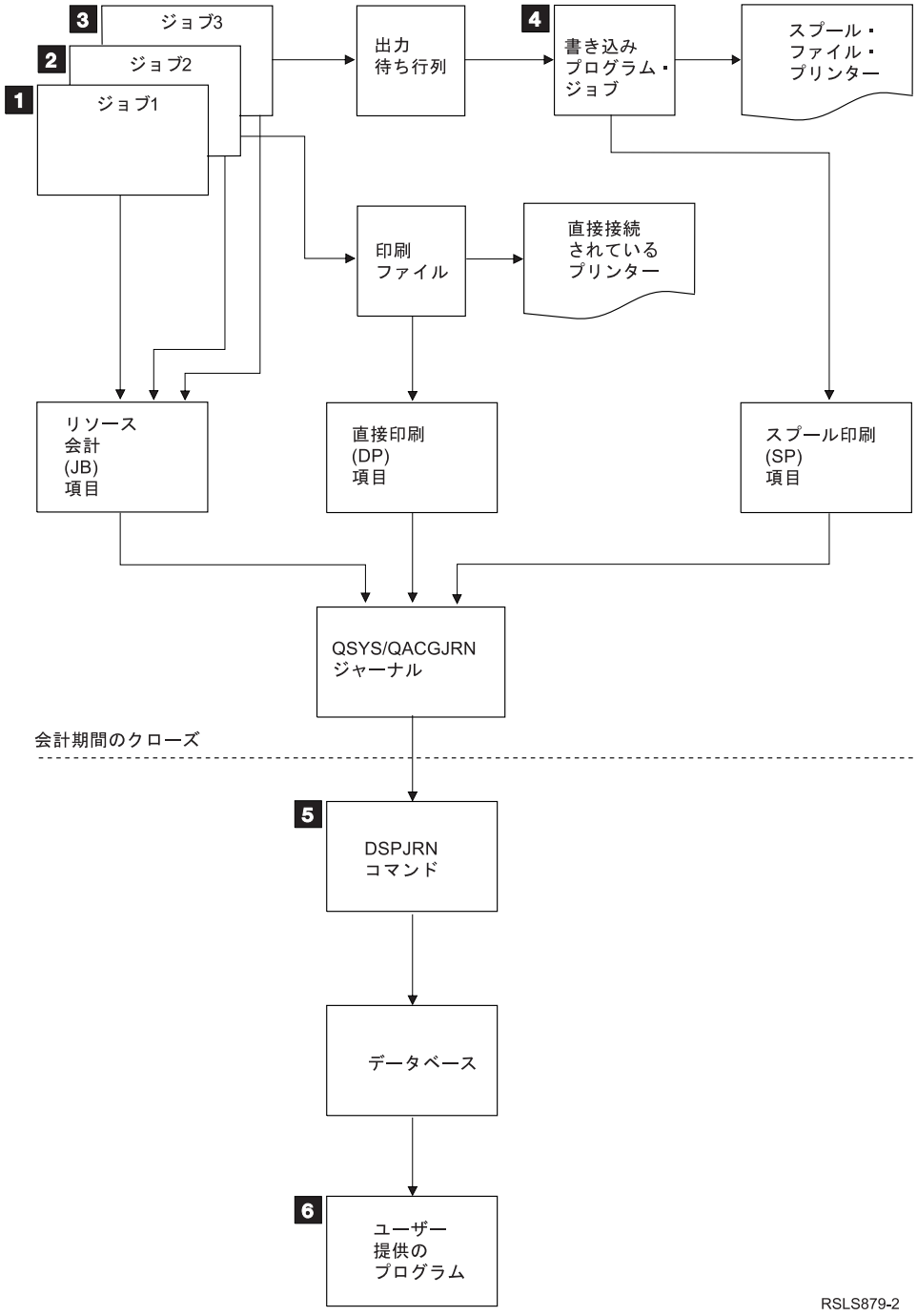


図 4. ジョブ会計の概説

1. ジョブ 1 が完了すると、システムは使用されたリソースを要約し、QACGJRN ジャーナルに JB ジャーナル項目を書き出します。ジョブの際に会計コードが変更された場合には、その変更のたびにおよびジョブの終了時に JB ジャーナル項目が書き出されます。ジョブ 1 はプリンター出力を全く作成しませんし、ジョブ・ログも作成されません。ですから、直接印刷 (DP) またはスプール印刷 (SP) ジャーナル項目は、ジョブ 1 に対しては作成されません。
2. ジョブ 2 は、プリンターにファイルを直接印刷します。ファイルが完了すると、印刷データを要約した DP ジャーナル項目が書き出されます。ジョブ 2 が完了すると、システムは使用したリソースを要約

し、JP ジャーナル項目を書き出します。ジョブ 2 はスプールされたプリンター出力を作成しません し、ジョブ・ログも作成されません。ですから、ジョブ 2 に対しては SP ジャーナル項目は作成されません。

3. ジョブ 3 は、スプールされたファイルに印刷します。SP ジャーナル項目は、印刷プログラムがファイルを印刷しない限りは書き出されません。 ジョブ 3 が完了すると、システムは使用したリソースを要約して、JB ジャーナル項目を書き出します。ジョブの完了時にジョブ・ログが作成される場合には、通常のスプール・ファイルと見なされ、ファイルの印刷時に SP ジャーナル項目が作成されます。
4. 印刷プログラムが開始され、1 つ以上のジョブにより作成されたファイルを印刷します。書き出しプログラムがファイルを完了すると、SP ジャーナル項目が作成されます。 SP ジャーナル項目は、印刷開始前にファイルがキャンセルされると作成されません。
5. 会計期間がクローズすると、ジャーナル表示 (DSPJRN) コマンドを使用して、累積されたジャーナル項目をデータベース・ファイルに書き込むことができます。
6. ユーザー作成プログラムまたは QUERY ユーティリティーを使用して、会計データを分析できます。使用したリソースなどの報告書では、特定の会計コード、ユーザー、またはジョブ・タイプ別にデータをコンパイルします。

ジョブ会計操作の特性:

システムは、主記憶域をできるだけ効率的に割り振ろうとします。ジョブは、実行されるたびに同じ量のリソースを使用するわけではないかもしれません。

たとえば、システム上にアクティブ状態にあるジョブがいくつかある場合、1 つのジョブがシステム環境を専用に使用している場合に比べ、必要なリソースを再確立するのにより多くに時間を費やすことになります。システムは、主記憶域の管理を支援するため、様々なジョブに割り当てられているジョブ優先順位および実行優先順位を使用します。ですから、優先順位の高いジョブは優先順位の低いジョブに比べて、少ないシステム・リソースしか使用しません。

こうしたシステム操作特性を持っているため、収集するジョブ会計データに対して独自の変換処理やアルゴリズムを適用したい場合もあります。システム使用に対して課金する場合、優先順位の高いジョブ、システムのピーク時における作業、またはクリティカル・リソースの使用に対してはより多く課金したいと思われるかもしれません。

会計ジャーナル処理:

会計ジャーナル QSYS/QACGJRN は、他のジャーナルとして処理されます。分かりやすくするためにこのジャーナルを会計情報のためだけに保持することが推奨されていますが、ファイルをこのジャーナルに記録することもできます。

ジャーナル項目送信 (SNDJRNE) コマンドを使用して、他の項目をこのジャーナルに送信できます。 幾つかのジャーナルの使用に関連して操作上の他の考慮事項もありますが、QACGJRN ジャーナルでファイル項目を許可しない ことには利点があります。通常、QACGJRN ジャーナルを単独で制御するのは簡単なため、特定の会計期間のすべてのジョブ会計項目におけるジャーナル・レシーバー数を最少にして、新しいジャーナル・レシーバーを会計期間の最初に開始できます。また、システム項目はジャーナル QACGJRN に出力されます。これはジャーナル・コード J の項目で、IPL およびジャーナル・レシーバーで実行される一般操作 (たとえば、レシーバーの保管) に関連しています。

ジョブ会計項目

ジョブ会計項目は、システム値変更 (CHGSYSVAL) コマンドが有効になった後にシステムに入れられた後続のジョブ以降、ジャーナル・レシーバー内に置かれます。 ジョブの会計レベルは、システムに入れられ

る際に判別されます。ジャーナル会計情報 (QACGLVL) システム値がジョブの開始後に変更されると、そのジョブに実行されている会計タイプでは無効です。ファイルを作成したジョブが会計で操作され、システム値が *PRINT に設定されていると、直接印刷 (DP) およびスプール印刷 (SP) 項目が生じます。 会計レベルが *PRINT に設定された後にスプール・ファイルが印刷される場合、または会計レベルが変更される前にファイルを作成したジョブが開始された場合、こうしたスプール・ファイルに対してはジャーナル処理は実行されません。

ジョブ会計を使用するとき

ここで取り上げている方法は、ジョブ会計を使用するべきかどうか、またそれを使用するタイミングを判別する上で役に立ちます。

ジョブ会計によって提供される追加情報

ジョブ会計には、CPF1164 によって提供されるすべての情報に加えて以下の情報があります。

- 会計コード
- プログラムによって作成される印刷ファイル、行、およびページの数
- データベースの読み取り、書き込み、および更新操作の数
- 通信読み取りおよび書き込み操作の数
- 印刷される実際の行およびページ
- ジョブがアクティブ状態だった、および中断された時間
- 印刷装置に送信される制御情報および印刷データの実際のバイト数

以下の場合、ジョブ会計機能はジョブ会計統計をより効率的に収集します。

- 使用されるデータベース、印刷装置、および通信に関するリソース情報が重要な場合。
- 会計コードがユーザーまたはジョブに割り当てられる場合。
- 印刷出力の情報が重要な場合。
- ジョブ会計を、完了ジョブではなくジョブ内の会計セグメントに基づいて実行する必要のある場合。
- アクティブ状態および中断状態の時間情報が必要な場合。

以下の場合、QHST メッセージはジョブ会計統計をより効率的に収集します。

- ジャーナル処理を含め追加のオブジェクトの管理を行いたくない場合。
- QHST ログに自動的に送信される CPF1124 メッセージおよび CPF1164 メッセージで提供される以外のリソース情報を必要としない場合。
- 印刷会計情報を必要としない場合。

注: CPF1164 メッセージと JB ジャーナル項目に記録される統計では、正確に一致しない場合もあります。これは、主に次の 2 つの要因によります。(1) CPF1164 統計は、JP ジャーナル統計より少し前に記録されます。さらに (2) CPF1164 メッセージでは丸めは 1 度だけしか行われませんが、JP ジャーナルでは会計コードが変更されるたびに一部のフィールドで丸めが行われます。

セキュリティーおよびジョブ会計

機密保護担当者 (またはこの権限を与えられているプログラム) か *ALLOBJ および *SECADM 権限を持つユーザーのみが、ジャーナル会計情報 (QACGLVL) システム値を変更できます。

変更は、新しいジョブがシステムに入れられると有効になります。この制約事項により、ジョブ会計が有効で機密保護担当者がシステム IPL を実行する場合には、機密保護担当者のジョブに対して会計項目が書き込まれます。

ジョブ会計コードを割り当てる権限

ジョブ会計コードは、ユーザー・プロファイル作成 (CRTUSRPRF) コマンド、ユーザー・プロファイル変更 (CHGUSRPRF) コマンド、または会計コード変更 (CHGACGCDE) コマンドを使用する権限を持っている場合に限り割り当てることができます。 これにより、会計コードの使用が制限され、変更に対する妥当性検査の基礎が提供されることになります。

*SECADM 特殊権限を持つユーザーのみが、CRTUSRPRF コマンドおよび CHGUSRPRF コマンドを使用することが許可されます。しかし、機密保護担当者は CL プログラムを作成してこの権限を委任することができます。そのプログラムにより、機密保護担当者のプロファイルを他のユーザーが引き受け、ユーザー・プロファイルの ACGCDE パラメーターを変更することができます。その後、そのユーザーは 1 つ以上の CL プログラムに対する権限を有することが可能です。

ACGCDE パラメーターもジョブ記述オブジェクト内に存在しますが、デフォルトの *USRPRF 以外の値を入力するには CHGACGCDE コマンドを使用する権限がなければなりません。 CHGACGCDE は、PUBLIC 権限 *USE と一緒になっています。

CHGACGCDE コマンドに対する権限

ユーザーに会計コード変更 (CHGACGCDE) コマンドの使用を許可すると、ユーザーは以下の事柄が可能になります。

- ジョブ記述内の ACGCDE パラメーターの作成または変更。 (ジョブ記述を作成または変更する権限も必要になります。)
- ユーザーの現行ジョブにおける会計コードの変更。
- ユーザーが *JOBCTL 特殊権限も有している場合、独自の会計コードではなくジョブの会計コードの変更。

CL プログラムで CHGACGCDE コマンドを使用して、プログラム所有者の権限が適用される別のセキュリティーを提供できます。 そのようにすると、外部機能を実行するユーザーは、CHGACGCDE コマンドに対する直接的な権限がなくてもセキュリティーの対象となっている機能を実行することができます。

会計ジャーナルおよびそのレシーバーは、セキュリティーの観点から他のジャーナル・オブジェクトとして扱われます。会計ジャーナルおよびジャーナル・レシーバーに持たせるべき権限を決定する必要があります。

関連タスク

232 ページの『アカウンティング・コードの割り当ての制御』
すべてのデータ処理アプリケーションにとって重要な側面は、必ず正しい制御フィールドを指定することです。ジョブ・アカウンティング・コードの場合、これには正式なコードの存在を検査するだけでなく、どのユーザーが特定コードの使用を許可されているかをも検査する、複合の妥当性検査機能が必要となることがあります。

会計コードについて

ジョブの初期会計コード (長さは最大 15 文字) は、ジョブ記述またはジョブのユーザー・プロファイル内の ACGCDE (会計コード) パラメーターによって決定されます。

ジョブが開始されると、ジョブ記述がジョブに割り当てられます。ジョブ記述オブジェクトには、ACGCDE パラメーターの値が含まれます。デフォルトの *USRPRF が使用されると、ジョブのユーザー・プロファイル内の会計コードが使用されます。

注: ジョブ投入 (SBMJOB) コマンドを使用してジョブを開始すると、会計コードはサブミッターのジョブのものと同じになります。
会計コード変更 (CHGACGCDE) コマンドを使用すると、ジョブをシステムに入れた後に会計コードを変更できます。

CRTUSRPRF コマンドおよび CHGUSRPRF コマンドは、 ACGCDE パラメーターをサポートしています。 デフォルトは *BLANK です。特定のユーザーのすべての作業を 1 つの会計コードに記録する場合、変更が必要なのはユーザー・プロファイルのみです。CRTJOBD コマンドおよび CHGJOBD コマンドの ACGCDE パラメーターに会計コードを指定すると、特定のジョブ記述の会計コードを変更できます。さらに CHGACGCDE コマンドによって、単一のジョブで別の会計コードを使用することもできます。

ジョブ属性検索 (RTVJOBA) コマンドおよびジョブ属性を検索する API を使用すると、CL プログラム内の現行の会計コードにアクセスできます。

関連タスク

231 ページの『ジョブ会計のセットアップ』
ジョブ会計をセットアップするには、文字ベース・インターフェースを使用します。

232 ページの『アカウンティング・コードの割り当ての制御』
すべてのデータ処理アプリケーションにとって重要な側面は、必ず正しい制御フィールドを指定することです。ジョブ・アカウンティング・コードの場合、これには正式なコードの存在を検査するだけでなく、どのユーザーが特定コードの使用を許可されているかをも検査する、複合の妥当性検査機能が必要となることがあります。

リソース会計

ジョブ・リソース会計データは、ジョブ完了時にジョブ (JB) ジャーナル項目に要約されます。さらに、会計コード変更 (CHGACGCDE) コマンドが発生するたびに使用されるリソースを要約する JB ジャーナル項目を、システムは作成します。 JB ジャーナル項目には、以下が含まれます。

- 完全修飾ジョブ名
- 直前に終了した会計セグメントの会計コード
- 処理装置時間
- 経路指定ステップの数
- ジョブがシステムに入れられた日時
- ジョブが開始された日時
- トランザクション合計時間 (サービス時間、不適格時間、およびアクティブ状態時間を含む)
- すべての対話式ジョブに対するトランザクション数
- 補助入出力操作
- ジョブ・タイプ
- ジョブ完了コード
- スプールまたは直接印刷の場合、作成される印刷行、印刷ページ、およびファイル数
- データベース・ファイルの読み取り、書き込み、更新、および削除の数
- ICF ファイル読み取りおよび書き込み操作数

注: 一部のジョブ会計情報は、QHST ログ内にある CPF1124 メッセージおよび CPF1164 メッセージを使用しても利用できます。

リソース会計データ

ジャーナル項目を分析する際、ジャーナル項目が書き込まれる方法およびタイミングを理解するのは重要です。JB ジャーナル項目は、ジョブ会計コードが変更される際およびジョブが終了するときにジョブのジョブ会計ジャーナルに書き込まれます。ですから、1 つのジョブに複数のジャーナル項目が存在する場合もあります。

各リソース会計ジャーナル項目には、以前の会計コードが有効だった際に使用されたリソースに関する情報が含まれます。以下の例について考慮してください。

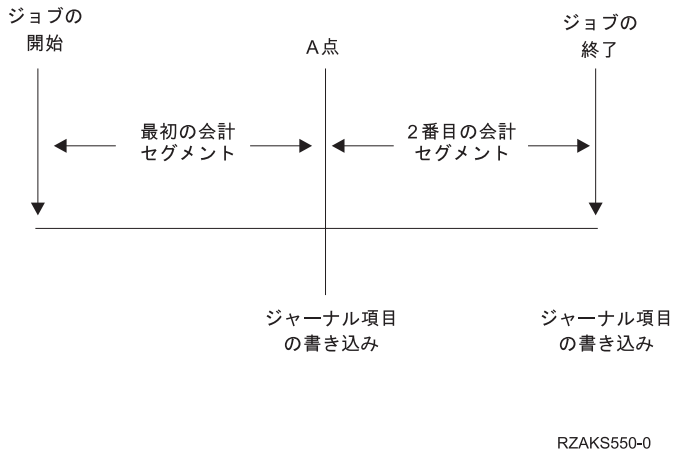


図 5. リソース会計データの例

A 点で、CHGACGCDE コマンドが発行されました。 会計コードが変更され、JB ジャーナル項目がジャーナルに送信されます。 JB ジャーナル項目には、最初の会計セグメントのデータが含まれます。ジョブが終了すると、そのジョブに対して 2 番目の会計セグメントのデータが含まれる 2 番目の JB 項目が作成されます。

ジョブが存在している間にジョブ会計コードが変更されなかった場合には、単一の JB 項目でジョブが使用したリソース合計について要約されます。ジョブが存在している間にジョブ会計コードが変更された場合には、複数の JB 項目のフィールドを合算して、ジョブが使用したリソース合計を判別する必要があります。JB 会計項目においては、ジョブ・ログの作成でジョブまたはその印刷出力の処理装置使用に関してはカウントされません。しかし、印刷ファイル会計を使用している場合、プリンター・ファイル・ジャーナル項目には印刷されるジョブ・ログが含まれます。

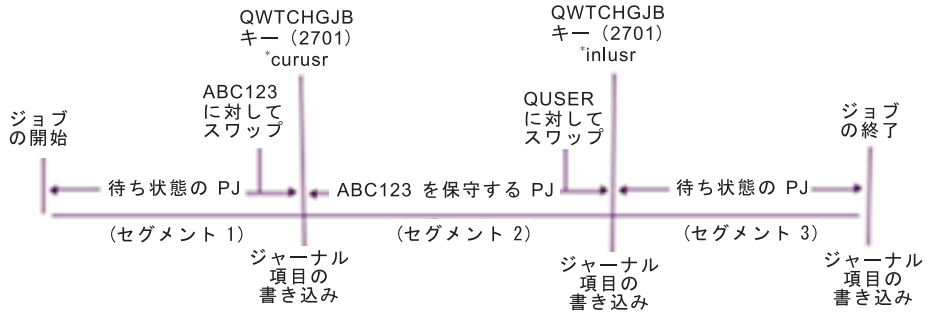
事前開始通信ジョブおよびジョブ会計

システムでジョブ会計を使用する場合、プログラム開始要求が事前開始ジョブに添付された直後に、事前開始ジョブの変更 (CHGPJ) コマンドを会計コード・パラメーター用のプログラム開始要求値 (CHGPJ ACGCDE(*PGMSTRRQS)) を使用して実行する必要があります。

このアクションにより、プログラム開始要求に関連するユーザー・プロファイル内で指定された値に会計コードが変更されます。プログラムがプログラム開始要求の処理を終える直前に、事前開始ジョブの変更コマンド (CHGPJ) を会計コード・パラメーター用の事前開始ジョブ項目値 (CHGPJ ACGCDE(*PJE)) を使用して実行してください。 これにより会計コードは、事前開始ジョブ項目のジョブ記述で指定された値に再び変更されます。

バッチ・アプリケーションの事前開始ジョブ

事前開始ジョブおよびそれを使用するサーバー・ジョブは通常、QUSER などの汎用ユーザー・プロファイルを使って開始するように構成されます。その後、要求が処理されるのを待ちます。処理すべき要求が事前開始ジョブに与えられると、ジョブはプロファイル処理の設定 (QWTSETP) API を使用してユーザー・プロファイルを要求元のユーザー・プロファイルに対してスワップし、要求を保守した後、再び最初のユーザー・プロファイルにスワップします。事前開始ジョブを再利用するように構成されている (事前開始ジョブ項目追加 (ADDPJE) または事前開始ジョブ項目変更 (CHGPJE) コマンドの MAXUSE パラメーターが 1 より大きい) 場合、ジョブは別の要求を待ち、上記のシナリオを繰り返します。この場合、1 つの事前開始ジョブが複数の異なるユーザーを保守できる場合があります。使用されるリソースに関してそれらの各ユーザーに課金できるようにする場合、毎回サービス要求を行う前と後に、会計コードを更新する必要があります。システム定義のサーバー・ジョブは最初からこれを行うよう設計されています。



以下は、書式設定に SQL または照会を使用した場合の、上記の図の 3 つのジャーナル項目の概要です。

表 3. 3 つの会計セグメントを持つ事前開始ジョブ

ジャーナル項目番号	ジョブ名	ジョブ・ユーザー	ジョブ番号	ユーザー・プロファイル	会計コード	CPU	トランザクション
1	QSVREX1	QUSER	123456	ABC123	QUSER	50	1
2	QSVREX1	QUSER	123456	QUSER	ABC123	3729	120
3	QSVREX1	QUSER	123456	QUSER	QUSER	73	2

使用されるリソース (例えば CPU やトランザクション) を会計コードに対して課金することはできますが、「ユーザー・プロファイル」フィールド (JAUSPF) の下にリストされているユーザーに対して常に課金できるとは限りません。ジャーナル項目が書き込まれた時点のユーザー・プロファイルが現行ユーザーとなりますが、それは必ずしも会計セグメント中ずっとアクティブだったユーザー・プロファイルであるとは限りません。この例で、ユーザー・プロファイルは最初の 2 つのセグメントにおいてそれぞれ 1 度スワップされました。ジャーナル項目はスワップ後に書き込まれているため、項目に記録される現行のユーザー・プロファイルは、前の会計セグメント中にリソースを使用したユーザーではありません。

同様にジョブ・ユーザーも、使用されるリソースの課金に関して信頼して使用できるものではありません。ジョブ・ユーザーはジョブ開始時のユーザーであり、修飾ジョブ名の一部として、別のユーザーを保守している場合でも、変更されないからです。会計コードは、リソースの使用に対する課金に関して唯一信頼して使用できるフィールドです。会計コードは変更されるまでジョブとともに保存されるため、他のユーザー・フィールドとは異なります。変更時にはまずジョブの現行の会計コードがジャーナル項目に書き込まれ、その後、新規会計コードがジョブに保管されます。

関連概念

51 ページの『事前開始ジョブ』
事前開始ジョブは、作業要求を受け取る前に実行を開始するバッチ・ジョブです。事前開始ジョブは、サブシステム内の他のタイプのジョブに先立って開始されます。事前開始ジョブは、事前開始ジョブ項目 (サブシステム記述の一部) を使用して、開始時に用いるプログラム、クラス、および記憶域プールを判別するので他のジョブとは異なります。

138 ページの『事前開始ジョブの管理』
プログラム開始要求を処理するのに必要な時間を減らすために、事前開始ジョブを使用することができます。これらは、実行可能な事前開始ジョブに関連した最も一般的なタスクです。

関連情報

Experience Report: 事前開始ジョブ項目の調整

経験報告: ジョブ・アカウンティング

ジョブ会計のシステム・ジョブ処理

制御するシステム・ジョブ (たとえば、読み取りプログラムおよび書き込みプログラム) には、*SYS の会計コードが割り当てられます。制御しない他のシステム・ジョブ (たとえば、QSYSARB、QLUS、SCPF) は、ジャーナル項目を受け取りません。

注: 会計コード変更 (CHGACGCDE) コマンドを使用して、サブシステム・モニターまたは読み取りプログラムや書き込みプログラムの会計コードを変更することはできません。 しかし、IBM 提供の該当するジョブ記述およびユーザー・プロファイルを変更してから再開すると、読み取りプログラムまたは書き込みプログラムの会計コードを変更できます。

バッチ処理およびジョブ会計

ジョブ投入 (SBMJOB) コマンドを使用して自動的に投入されるバッチ・ジョブは、そのバッチ・ジョブを投入したジョブと同じ会計コードを使用します。 SBMJOB コマンドが使用されると、ジョブ記述項目がコード化される方法にかかわりなく、会計コードは指定変更できません。

投入するジョブ以外の会計コードでバッチ・ジョブを操作したい場合、会計コード変更 (CHGACGCDE) コマンドを以下のいずれかの方法で発行してください。

- SBMJOB コマンドを発行する前後。
- バッチ・ジョブ発行直後。

読み取りプログラムまたはデータベース・ジョブ投入 (SBMDBJOB) コマンドを使用して投入されるバッチ・ジョブは、バッチ・ジョブのジョブ記述で指定される会計コードを使用します。 ジョブ記述が ACGCDE(*USRPRF) を指定する場合、会計コードはそのジョブで使用するユーザー・プロファイルから取られます。

対話式処理およびジョブ会計

対話式ジョブにユーザー用に修正されたオプションがあり、それぞれのオプションに会計コードが割り当てられている場合には、ユーザーが新しい機能を使って作業することを要求する際に自動的に新規コードが割り当てられるのが望ましい場合があります。

一般的なのは、メニュー・オプションで新しい機能領域を要求するという方法です。会計コード変更 (CHGACGCDE) コマンドが制御言語プログラムで発行され、以前の会計コードで使用されたジョブ値が JB 会計ジャーナル項目に要約されます。

使用する会計コードをユーザーだけが把握している割り当てがいくつかある場合、以下のようにできます。

- ユーザーに、CHGACGCDE コマンドを入力する権限を与えます。

- 会計コードに関するプロンプトをユーザーに出すプログラムを作成します。

注: ソース・パススルー・ジョブの場合、ジョブ会計情報にはターゲット・パススルー・ジョブは含まれません。ターゲット・パススルー・ジョブの場合、ジョブ会計情報には関連する通信バッチ・ジョブは含まれません。

プリンター・ファイル会計

プリンター・ファイル会計には、非スプール・プリンター・ファイル用 DP とスプール・プリンター・ファイル用 SP の 2 つのタイプのジャーナル項目があります。 一部の情報は SP 項目でのみ使用可能ですが、これら 2 つのタイプのジャーナル項目では共通ジャーナル項目フォーマットを共用しています。 DP および SP ジャーナル項目には、以下のような情報が含まれます。

- 完全修飾ジョブ名
- 会計コード
- 装置ファイル名およびライブラリー
- 装置名
- 装置タイプおよび型
- 印刷ページ総数および印刷行総数。複数部が印刷される場合、すべての部数の合計になります。
- スプール・ファイル名 (SP 項目のみ)
- スプール・ファイル番号 (SP 項目のみ)
- 出力優先順位 (SP 項目のみ)
- 元のタイプ (SP 項目のみ)
- 用紙タイプ (SP 項目のみ)
- 印刷装置に送信される制御情報および印刷データのバイト総数。複数部が印刷される場合、すべての部数の合計になります。(SP 項目にのみ適用されます。)

DP および SP ジャーナル項目は、ファイルの印刷時に生じます。 スプール・ファイルが印刷されないと、SP ジャーナル項目は生じません。

ジョブ会計のジャーナル項目

システムには、収集可能な多様なタイプのデータ用の種々のジャーナル項目が備えられています。

- ジョブ・リソース会計: ジョブ (JB) ジャーナル項目には、ジョブで使用するリソース、またはジョブ内で用いる種々の会計コードに使用するリソースを要約したデータが含まれます。
- プリンター・ファイル会計:
  - 直接印刷 (DP) ジャーナル項目: 印刷装置 (非スプール) で生成されたプリンター・ファイルに関するデータが含まれます。
  - スプール印刷 (SP) ジャーナル項目: 印刷プログラム (スプール) によって作成されるプリンター・ファイルに関するデータが含まれます。

ジョブ会計ジャーナル項目のフィールド情報:

これらの表は、ジョブ・ジャーナル項目にあるフィールド情報をリストしています。様々なフィールドに関する追加情報は、フィールド参照ファイル QSYS/QAJBACG および QSYS/QAJBACG4 に記されています。

表 4. ジョブ・ジャーナル項目のフィールド

フィールド名 (14 文字)	説明	フィールド属性	コメント
JAJOB	ジョブ名	Character (10)	
JAUSER	ジョブ・ユーザー	Character (10)	
JANBR	ジョブ番号	Zoned (6,0)	
JACDE	会計コード	Character (15)	
JACPU	使用された処理装置時間 (ミリ秒単位)	Packed decimal (11,0)	処理装置時間には、ジョブ・ログの作成に関する処理装置使用統計および印刷装置統計は含まれません。
JARTGS	経路指定ステップの数	Packed decimal (5,0)	
JAEDTE	システムに入れられたジョブ - ジョブ入力日付 (mmddyy フォーマット)	Character (6)	
JAETIM	システムに入れられたジョブ - ジョブ入力時刻 (hhmmss フォーマット)	Character (6)	
JASDTE	ジョブ開始日時 - ジョブ開始日 (mmddyy フォーマット)	Character (6)	ジャーナル項目のジョブ完了日時に関しては、標準ジャーナル項目の接頭部情報の一部である JODATE および JOTIME フィールドを使用します。(こうしたフィールドに関する詳細は「バックアップおよび回復の手引き」を参照してください。)システムの異常終了後には、こうしたフィールドには現行日時が含まれ、(CPF1164 メッセージのように) システム終了の実際の時間は含まれません。
JASTIM	ジョブ開始日時 - ジョブ開始時間 (hhmmss フォーマット)	Character (6)	ジャーナル項目のジョブ完了日時に関しては、標準ジャーナル項目の接頭部情報の一部である JODATE および JOTIME フィールドを使用します。(こうしたフィールドに関する詳細は「バックアップおよび回復の手引き」を参照してください。)システムの異常終了後には、こうしたフィールドには現行日時が含まれ、(CPF1164 メッセージのように) システム終了の実際の時間は含まれません。
JATRNT	合計トランザクション時間 (秒単位)	Packed decimal (11,0)	以下の場合には、合計トランザクション時間は -1 に設定されます。 • 時間が過去に遡って設定される。 • ファイル内の計算においてオーバーフローが生じた。 • ジョブがアクティブ状態にあるときにシステムがダウンした。
JATRNS	トランザクションの数	Packed decimal (11,0)	最終トランザクション (SIGNOFF) はカウントされません。

表 4. ジョブ・ジャーナル項目のフィールド (続き)

フィールド名 (14 文字)	説明	フィールド属性	コメント
JAAUX	同期補助入出力操作およびデータベース操作 (何らかの理由によるページ不在を含みます)	Packed decimal (11,0)	
JATYPE	ジョブ・タイプ	Character (1)	記録されるジョブ・タイプは以下のとおりです。 **A** 自動開始ジョブ **B** バッチ・ジョブ (通信および MRT を含む) **I** 対話式ジョブ **M** サブシステム・モニター **R** スプール読み取りプログラム **W** スプール書き出しプログラム **注:** これらは、メッセージ CPF1164 で使用されるのと同じです。ただし、ジャーナル項目には含まれずに、一部のシステム・ジョブ情報を含むメッセージ CPF1164 は例外です。
JACCDE	完了コード	Packed decimal (3,0)	メッセージ CPF1164 で使用されるのと同じ完了コードです。以下のとおりです。 **000** 正常完了 **010** 制御された終了時またはサブシステムの制御された終了時の正常完了 **020** ジョブが終了重大度を超えました。 **030** ジョブが異常終了しました。 **040** ジョブがアクティブになる前に終了しました。 **050** ジョブがアクティブなときに終了しました。 **060** ジョブがアクティブなときにサブシステムが終了しました。 **070** ジョブがアクティブなときにシステムが異常終了しました。 **080** 時間制限内にジョブが完了しました。 **090** 時間制限の終了後にジョブは強制的に完了しました。 **099** CHGACGCDE コマンドによる会計項目
JALINE	印刷行数	Packed decimal (11,0)	印刷行数は、実際に印刷される行数を示すものではありません。スプール・ファイルをキャンセルできますし、または複数部を印刷できます。JB ジャーナル項目内のこの情報は、プログラムが書き出した行数をのみ示します。ジョブ・ログに書き出される行数は除外されます。この章の後続部分で取り上げられる DP および SP プリンター・ファイルの会計データに関して参照してください。
JAPAGE	印刷ページ数	Packed decimal (11,0)	
JAPRTF	印刷ファイル数	Packed decimal (11,0)	

表 4. ジョブ・ジャーナル項目のフィールド (続き)

フィールド名 (14 文字)	説明	フィールド属性	コメント
JADBPT	データベース書き込み操作の数	Packed decimal (11,0)	記録されるデータベース入出力操作の数には、読み取りプログラムおよび書き込みプログラムへの入出力操作、または CL コマンド CPYSPLF、DSPSPLF、または WRKSPLF による入出力操作は含まれません。SEQONLY(*YES) が有効の場合、この数は読み取られた個々のレコード数ではなく、読み取られたレコードの各ブロック数を示します。
JADBGT	データベース読み取り操作の数	Packed decimal (11,0)	記録されるデータベース入出力操作の数には、読み取りプログラムおよび書き込みプログラムへの入出力操作、または CL コマンド CPYSPLF、DSPSPLF、または WRKSPLF による入出力操作は含まれません。SEQONLY(*YES) が有効の場合、この数は読み取られた個々のレコード数ではなく、読み取られたレコードの各ブロック数を示します。
JADBUP	データベース更新、FEOD の削除、解放、コミット、およびロールバック操作の数	Packed decimal (11,0)	記録されるデータベース入出力操作の数には、読み取りプログラムおよび書き込みプログラムへの入出力操作、または CL コマンド CPYSPLF、DSPSPLF、または WRKSPLF による入出力操作は含まれません。SEQONLY(*YES) が有効の場合、この数は読み取られた個々のレコード数ではなく、読み取られたレコードの各ブロック数を示します。
JACMPT	通信書き込み操作の数	Packed decimal (11,0)	記録される通信入出力操作の数には、リモート・ワークステーション・アクティビティーは含まれません。通信装置用の入出力の場合、この数には ICF ファイルに関連したアクティビティーのみが含まれます。
JACMGT	通信読み取り操作の数	Packed decimal (11,0)	記録される通信入出力操作の数には、リモート・ワークステーション・アクティビティーは含まれません。通信装置用の入出力の場合、この数には ICF ファイルに関連したアクティビティーのみが含まれます。
JAACT	ジョブがアクティブだった時間 (ミリ秒単位)	Packed decimal (11,0)	
JASPN	ジョブが中断状態にあった時間 (ミリ秒単位)	Packed decimal (11,0)	
JAEDTL	ジョブがシステムに入れられたタイム・スタンプ (mmddyyyyhhmmss)	Character (14)	

表 4. ジョブ・ジャーナル項目のフィールド (続き)

フィールド名 (14 文字)	説明	フィールド属性	コメント
JAESTL	ジョブを開始したタイム・スタンプ (mmddyyyyhhmmss)	Character (14)	
JAAIO	データベースおよび非データベース操作の非同期入出力	Packed decimal (11,0)	
JAXCPU	使用された拡張 CPU 時間	Packed decimal (29,0)	
JAXSIO	拡張された同期補助入出力操作	Packed decimal (29,0)	
JAXAIO	拡張された非同期補助入出力操作	Packed decimal (29,0)	
JAXDBP	データベース書き込みの拡張数	Packed decimal (29,0)	
JAXDBG	データベース取得の拡張数	Packed decimal (29,0)	
JAXDBU	データベース更新および削除の拡張数	Packed decimal (29,0)	
JAXLIN	印刷の拡張行数	Packed decimal (29,0)	
JAXPAG	印刷ページの拡張数	Packed decimal (29,0)	
JAXPRT	印刷ファイル数	Packed decimal (29,0)	

直接印刷およびスプール印刷のためのプリンター・ファイルの会計データ:

直接印刷 (DP) またはスプール印刷 (SP) ジャーナル項目に使用される会計コードは、ファイルがクローズされる際のジョブの会計コードです。時折 DP または SP 項目は、ファイルがクローズする前 (SCHEDULE(*IMMED) ファイルを作成した書き出しプログラムが終了するときなど) に作成されます。この場合、ジョブの現行の会計コードが使用されます。

DP または SP ジャーナル項目は、印刷される各ファイルごとに作成されます。 ジョブ・ログがスプールされ、その後印刷される場合、SP 項目がそのために作成されます。 さらに、SP 項目が、印刷プログラムによって、プリンターにリダイレクトされるディスケット・スプール・ファイルに書き出されます。

DP 会計ジャーナル情報:

QSYS/QAPTACG5 ファイルには、直接印刷 (DP) ジャーナル項目で使用されるフィールドが含まれます。この表には、こうしたフィールドやその属性がリストされています。

表 5. 直接印刷ジャーナル項目のフィールド

フィールド名	説明	フィールド属性
JAJOB	ジョブ名	Character (10)
JAUSER	ジョブ・ユーザー	Character (10)
JANBR	ジョブ番号	Zoned (6,0)
JACDE	会計コード	Character (15)

表 5. 直接印刷ジャーナル項目のフィールド (続き)

フィールド名	説明	フィールド属性
JADFN	装置ファイル名	Character (10)
JADFNL	装置ファイルが保管されるライブラリー	Character (10)
JADEVN	装置名	Character (10)
JADEVT	装置タイプ	Character (4)
JADEVM	装置モデル	Character (4)
JATPAG	生成される印刷ページ総数	Packed decimal (11,0)
JATLIN	生成される印刷行総数	Packed decimal (11,0)
JASPFN	常時ブランク	Character (10)
JASPNB	常時ブランク	Character (4)
JAOPTY	常時ブランク	Character (1)
JAFMTP	常時ブランク	Character (10)
JABYTE	常時ゼロ	Packed decimal (15,0)
JAUSRD	ユーザー・データ	Character (10)
JALSPN	常時ブランク	Character (6)
JASPSY	常時ブランク	Character (8)
JASPDT	常時ブランク	Character (7)
JASPTM	常時ブランク	Character (6)
JADFASP	常時ブランク	Character (10)

SP 会計ジャーナル情報:

この表は、スプール印刷 (SP) ジャーナル項目内で使用されるフィールド (QSYS/QAPTACG5 ファイル内に存在) をリストしています。

注: SP 会計ジャーナル情報は、スプール・ファイル名、スプール・ファイル番号、出力優先順位、用紙タイプ、およびプリンターに送信される制御情報および印刷データのバイト総数が含まれているという点を除いては、直接印刷 (DP) 会計ジャーナル・データで提供されている情報と似ています。 SP ジャーナル項目は、書き込みプログラムが装置に対するファイルの書き込みを開始する前にスプール・ファイルが削除されると、書き込まれません。

表 6. スプール印刷ジャーナル項目のフィールド

フィールド名	説明	フィールド属性
JAJOB	ジョブ名	Character (10)
JAUSER	ジョブ・ユーザー	Character (10)
JANBR	ジョブ番号	Zoned (6,0)
JACDE	会計コード	Character (15)
JADFN	装置ファイル名	Character (10)
JADFNL	装置ファイルが保管されるライブラリー	Character (10)
JADEVN	装置名	Character (10)
JADEVT	装置タイプ	Character (4)

表 6. スプール印刷ジャーナル項目のフィールド (続き)

フィールド名	説明	フィールド属性
JADEVM	装置モデル	Character (4)
JATPAG	生成される印刷ページ総数	Packed decimal (11,0)
JATLIN	生成される印刷行総数	Packed decimal (11,0)
JASPFN	スプール・ファイル名	Character (10)
JASPNB	スプール・ファイル番号	Character (4)
JAOPTY	出力優先順位	Character (1)
JAFMTP	用紙タイプ	Character (10)
JABYTE	プリンターに送信されるバイト総数	Packed decimal (15,0)
JAUSRD	ユーザー・データ	Character (10)
JALSPN	スプール・ファイル番号	Character (6)
JASPSY	スプール・ファイルのジョブ・システム名	Character (8)
JASPDT	スプール・ファイルの作成日 (cyymmdd フォーマット)	Character (7)
JASPTM	スプール・ファイルの作成時刻 (hhmmss フォーマット)	Character (6)
JADFASP	装置ファイル・ライブラリーの ASP 名	Character (10)

注:

- システムは実際に印刷されたページ数、行数、およびバイト数を記録しようとしますが、書き込みプログラムが *IMMED をキャンセルしたり、装置エラー (用紙切れなど) から回復すると、印刷された実際のページ数、行数、およびバイト数を判別するのは不可能です。
- 位置合わせ行のある追加ページおよび行は、カウントされるページ、行、およびバイトには含まれません。
- スプール・ファイルの状況が (MSGW に設定されているものの) WTR になるか、MSGW 状況の間にファイルが削除される場合、 SP ジャーナル項目は DP 会計ジャーナルに出力され、印刷されたページが 0 (ゼロ)、行も 0 (ゼロ)であることを示します。
- 印刷装置構成 AFP(*YES) を使用している際、ページを印刷した直後のファイルを削除または保留すると、実際には数ページ印刷されたとしても、そのファイルの SP 項目では印刷されたページが 0 (ゼロ) で、行も 0 (ゼロ) であると示されます。
- ジョブおよびファイル区切りでカウントされるページ、行、およびバイトは、関連するファイルのカウントに組み込まれます。
- IPDS ファイルにグラフィックスまたはバーコードが含まれ、グラフィックスやバーコードをサポートしていない IPDS 印刷装置に送信される場合、ページ、行、およびバイトのカウントには印刷されないグラフィックスおよびバーコードが含まれます。
- 印刷装置構成が AFP(*YES) の場合には、生成される印刷行総数のフィールドはゼロになります。生成されるページ総数のフィールドは正確です。

作業の管理

システム・オペレーターまたは管理者としての作業の 1 つは、サーバーの稼働状況を良好に保つということです。これは、ジョブ、ジョブ待ち行列、サブシステム、メモリー・プール、ジョブ・ログ、および出力待ち行列機能をモニターし、管理し、適正な状態に保つということを意味します。

このセクションでは、さまざまな種類の日次実行管理タスクと、システムで実行する必要があるかもしれない他のタスクについて、トピックとして取り上げます。それぞれのサブトピックでは、それらのタスクを実行することが重要である理由、およびそれらのタスクを完了させる方法について説明します。

特殊 IPL 回復プログラムの呼び出し

直前のシステム終了が異常であったことを IPL が感知する場合に特殊回復プログラムを呼び出すには、自動開始ジョブ項目を制御サブシステムのサブシステム記述に追加します。

このプログラムは、直前のシステム終了状況 (QABNORMSW) システム値をチェックします。正常なシステム終了の場合は QABNORMSW の値は '0'、異常なシステム終了の場合は QABNORMSW の値は '1'です。代替手段として、リカバリー機能の完了時にメッセージをドロップして他のサブシステムを開始するという方法があります。

```
1.00 /* SPCRECOV - 特殊回復プログラムを呼び出す自動開始プログラム */
2.00                    PGM
3.00                    DCL      &QABNORMSW *CHAR LEN(1)
4.00                    RTVSYSVAL SYSVAL(QABNORMSW) RTNVAR(&QABNORMSW)
5.00                    IF       (&QABNORMSW *EQ '1') DO /* Recover */
6.00                    SNDPGMMSG MSG('回復プログラム実行中 - 指示があるまで +
7.00                                サブシステムを開始しないでください。') +
8.00                                TOMSGQ(QSYSOPR)
9.00                    CALL      RECOVERY
10.00                    SNDPGMMSG MSG('回復完了 - ジョブを開始できます。') +
11.00                                TOMSGQ(QSYSOPR)
12.00                     ENDDO /* Recover */
13.00                     ENDPGM
```

関連情報

IPL 始動プログラムの変更

システム・アクティビティーのモニター

管理者が一日に行うさまざまな重要な仕事の一つに、システム・アクティビティーのモニターがあります。システムを通過する作業のフローのモニターは、毎日モニターしなければならない情報のほんの一部に過ぎません。これは数種類の方法で実行できます。たとえば、System i ナビゲーター や System i ナビゲーター マネージメント・セントラルを使用することができます。

文字ベース・インターフェースのシステム状況の処理 (WRKSYSSTS) 画面の上半分をモデルにした「システム状況」ウィンドウで、システムの状況を迅速に、かつ簡単に検査できます。マネージメント・セントラルを使用すると、システム・モニターを介して、より詳細な機能をモニターすることができます。

「**システム**」フォルダーまたは「**実行管理機能**」フォルダーから「システム状況」ウィンドウにアクセスできます。

「**システム**」フォルダーからシステム状況を調べるには、次のようにします。

1. System i ナビゲーター で、「**ユーザー接続**」を展開します。
2. 処理したい接続を右マウス・ボタン・クリックし、「**システム状況**」をクリックします。

「実行管理機能」フォルダーからシステム状況を入手するには、次のようにします。

1. System i ナビゲーターで、「**実行管理機能**」を展開します。
2. 「**実行管理機能**」を右マウス・ボタン・クリックして、「**システム状況**」をクリックします。

システム状況を使って実行できるさまざまなタスクの詳細については、System i ナビゲーター のヘルプを参照してください。

メモリー・プールの使用状況の検査

メモリー・プールが使用するメモリーのサイズを定期的に検査することは重要です。これらのレベルをモニターすることによって、プールが効率よく実行するように調整でき、作業の実行サイクルを円滑に保つことができます。 System i ナビゲーター を使用して、プールが使用しているメモリーのサイズを簡単にモニターすることができます。

メモリーの使用状況を調べるには、以下のステップに従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 処理するメモリー・プール (たとえば「対話式」) を右マウス・ボタン・クリックして、「**プロパティー**」を右マウス・ボタン・クリックします。
3. 「構成」タブをクリックします。「サイズ」グループ内にある「**現行**」フィールドには、プールが現在使用しているメモリーのサイズが表示されます。

   注: メモリー・プールの現行サイズは、「**アクティブ・プール**」または「**共用プール**」をクリックすることによっても表示できます。「現行」サイズ (メガバイト) は、メモリー・プールのリストを System i ナビゲーター の右側のペインに表示する場合のデフォルト列です。

システム・アクティビティーのレベルの制御

システム上のアクティビティーの数の制御は、サブシステム内で同時にいくつのジョブをアクティブにできるかの制御、または開始済みのジョブによる処理装置の使用の制御によって実行できます。

表 7. システム・アクティビティー・レベルの制御方法

制御できる対象	制御に使用できるもの	文字ベース・インターフェースによる方法	System i ナビゲーター・インターフェースによる方法
アクティブ・ジョブの数	サブシステム記述	コマンド: CHGSBSD MAXJOBS サブシステム内で同時にアクティブにできるジョブの数を指定するには、このパラメーターを使用します。 アクティブ・サブシステムの場合、サブシステム内で実行処理項目により開始された、同時にアクティブにできるすべてのジョブの合計は、MAXJOBS パラメーター値を超えることはできません。 これは自動開始ジョブは除外します。自動開始ジョブにより、サブシステムの開始時にこの制限を一時的に超える場合があります。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGSBSD コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
	ジョブ待ち行列項目	コマンド: CHGJOBQE MAXACT サブシステム内で同時にアクティブにできるジョブ待ち行列からのバッチ・ジョブの数を指定するには、このパラメーターを使用します。 ジョブ待ち行列に対する MAXACT に 1 を指定すると、ジョブ待ち行列からのジョブ優先順位によるジョブの順次選択が強制されます。 指定されたジョブ優先順位でアクティブにできるジョブの数を指定するには、MAXPTYn パラメーターを使用します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGJOBQE コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
	ワークステーション項目	コマンド: CHGWSE MAXACT WRKSTNTYPE パラメーターを指定する場合は、このパラメーターを使用します。このパラメーターは、その項目のサブシステム内で同時にアクティブにできる対話式ジョブの数を指定します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGWSE コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
	通信項目	コマンド: CHGCMNE MAXACT その項目で同時にアクティブにできる通信バッチ・ジョブの数を指定するには、このパラメーターを使用します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGCMNE コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
	経路指定項目	コマンド: CHGRTGE MAXACT 特定の経路指定項目を使用して同時にアクティブにできるジョブの数を指定するには、このコマンドを使用します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGRTGE コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
	事前開始ジョブ項目	コマンド: CHGPJE MAXJOBS その項目で同時にアクティブにできる事前開始ジョブの数を指定するには、このパラメーターを使用します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGPJE コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。
アクティブ・ジョブの数 (続き)	システム	最大適格スレッド (QMAXACTLVL) システム値は、主記憶域およびプロセッサー・リソースを同時に共用できるスレッドの数を指定するために使用します。すべての記憶域プール内のすべてのアクティブ・ジョブ (システム・ジョブを含む) は、QMAXACTLVL により制御されます。	**「ユーザー接続」 → 「サーバー」 → 「構成およびサービス (Configuration and Service)」 → 「システム値 (System Values)」 → 「パフォーマンス・カテゴリー (Performance category)」 → 「メモリー・プール」タブ → 「最大適格スレッド (Maximum eligible threads)」**

表 7. システム・アクティビティー・レベルの制御方法 (続き)

制御できる対象	制御に使用できるもの	文字ベース・インターフェースによる方法	System i ナビゲーター・インターフェースによる方法
処理装置および主記憶域の使用	基本記憶域プール	基本メモリー・プールの最大適格スレッド (QBASACTLVL) システム値は、同時に基本記憶域プールを共用できるスレッドの数を指定し、主記憶域の競合を制限するために使用します。	**「ユーザー接続」 → 「サーバー」 → 「構成およびサービス (Configuration and Service)」 → 「システム値 (System Values)」 → 「パフォーマンス・カテゴリー (Performance category)」 → 「メモリー・プール」タブ → 「基本メモリー・プール: 最大適格スレッド (Base Memory pool: Maximum eligible threads)」**
	共用プール	コマンド: WRKSHRPOOL 共用プールのアクティビティー・レベルを指定するには、このコマンドを使用します。	**「ユーザー接続」 → 「サーバー」 → 「実行管理機能」 → 「メモリー・プール」 → 「共用プール」 → 共用プールを右マウス・ボタン・クリック → 「プロパティー」 → 「構成」タブ**の順に展開し、**「最大適格スレッド (Maximum eligible threads)」**フィールドを変更する。
	専用記憶域プール	コマンド: CHGSBSD POOLS ユーザー定義の主記憶域プールのアクティビティー・レベルを指定するには、このコマンドを使用します。	「コマンドの実行」ウィンドウを使用します。 **エンドポイント・システムを右マウス・ボタン・クリックし、 → 「コマンドの実行」**を開きます。 CHGSBSD コマンドを入力し、**「プロンプト (Prompt)」**をクリックします。

例: アクティビティー制御の関係:

これらの例は、いくつかのアクティビティー制御の関係を示しています。システム・アクティビティー・レベルが 100 であり、ジョブが単一スレッドであるとします。

基本メモリー・プールの例

2 つのサブシステム SBSA および SBSB が、ジョブの実行に基本メモリー・プールを使用しています。SBSA には現在、このメモリー・プールで実行している 2 つのジョブがあり、SBSB には 1 つがあります。 SBSB のサブシステム記述内のジョブ待ち行列項目は、任意の数のジョブを開始できることを指定します。基本メモリー・プールのアクティビティー・レベルは 3 です。したがって、基本メモリー・プール内の 3 つのジョブだけが、処理装置について同時に競合する可能性があります。ただし、すべてのジョブは開始します。

サブシステム内に 4 つのジョブがある例

1 つの自動開始ジョブ、2 つのワークステーション・ジョブ、および 1 つのバッチ・ジョブ (合計で 4 つのジョブ) がサブシステム SBSC 内にあります。 SBSC の MAXACT は 4 と指定されています。実行処理項目の MAXACT がどのように指定されているとしても、少なくとも 1 つのジョブが実行を完了するまで、他のジョブは開始できません。

バッチ・サブシステム MAXACT(1) の例

サブシステム SBSE は、MAXACT に 1 が指定されているバッチ・サブシステムです。ジョブ待ち行列項目が MAXACT を指定していないとしても、サブシステムの MAXACT に 1 が指定されているので、制限は 1 ジョブです。したがって、ジョブはジョブ待ち行列から出ると一度に 1 つずつ、ジョブ優先順位で処理されます。

ジョブの状況の判別

ジョブをモニターすることによって、ジョブの状況を理解することができます。 ジョブ状況は、ジョブが何を行っているのかを調べるのに使用できる情報の重要な部分です。

アクティブ・ジョブまたはサーバー・ジョブの状況を調べるには、以下の指示に従います。

1. System i ナビゲーターで、**「ユーザー接続」 → *ご使用のシステム* → 「実行管理機能」 → 「アクティブ・ジョブ」**または**「サーバー・ジョブ」**の順に展開します。

   注: 「実行管理機能」フォルダーのどの場所からでも、ジョブにアクセスしてジョブ状況を確認することができます。
2. 「詳細状況」列を調べて、ジョブの状況 (たとえば、イベントを待機中、時間間隔を待機中、または待ち行列解除を待機中など) を判別します。

ヒント: 「詳細状況」列が表示されない場合、**「アクティブ・ジョブ」**(または**「サーバー・ジョブ」**) を右マウス・ボタン・クリックして、**「この表示をカスタマイズ (Customize this view)」 → 「列 (Columns)」**を選択して、画面に追加することができます。

サブシステムのモニター

サブシステムはシステム上で実行される日常のアクティビティーにとって重要なので、サブシステム内のアクティビティーをモニターすることは重要です。

サブシステム記述で、最大アクティブ・ジョブ数値を設定することによって、サブシステム内で一度に実行できるジョブの数を指定できます。システム上の作業量が増えるにつれて、サブシステム内の最大アクティブ・ジョブ数値を変更することもできます。ここに入力する数は、利用可能なリソースを適切に使用できるように設定する必要があります。使用できるリソースを増やさずにアクティブ・ジョブの数を増やしてしまうと、システムのパフォーマンスを損ないかねません。

使用しているサブシステムの最大アクティブ・ジョブ数値を確認するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

System i ナビゲーター:

1. System i ナビゲーターで、**「ユーザー接続」 → *ご使用のシステム* → 「実行管理機能」 → 「サブシステム」 → 「アクティブ・サブシステム」**の順に展開します。
2. モニターするサブシステムを右マウス・ボタン・クリックします。
3. **「プロパティー」**を選択します。

   注: このオプションの設定は必ず注意して行ってください。最大アクティブ・ジョブ数値に設定する値が大き過ぎると、システムの実行速度が遅くなります。一方、最大アクティブ・ジョブ数値に設定する値が小さ過ぎると、作業がボトルネックに引っ掛かってパフォーマンスが遅くなってしまいます。

文字ベース・インターフェース:

コマンド: サブシステム記述表示 (DSPSBSD)

オプション 1 (操作属性) を選択して、サブシステム内の最大ジョブ数の値を表示します。

メモリー・プールを使用したサブシステム数の判別

複数のジョブを実行するために、サブシステムには特定の比率でメモリーが割り振られます。いくつの異なるサブシステムが同じメモリー・プールからプルしているかを知っていることは重要です。いくつのサブシステムがプールにジョブを送り出しているか、またいくつのジョブがプールで実行されているかを知ると、プールのサイズとアクティビティー・レベルを調整して、リソースの競合を減らしたいと場合もあります。

System i ナビゲーター:

System i ナビゲーター を使用して、メモリー・プールを使用しているサブシステムの数をモニターするには、以下のようにします。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (Connection)**」 → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 処理するメモリー・プールを右マウス・ボタン・クリックして、「**サブシステム**」をクリックします。

   このウィンドウから、同一のメモリーを使用してジョブを実行しているサブシステムの数を判断することができます。

文字ベース・インターフェース:
コマンド: サブシステム処理 (WRKSBS)

このコマンドは、すべてのサブシステムおよびその対応するプールのリストを表示します。

ジョブ・パフォーマンス統計の表示

あるジョブの実行状況が極端に悪いと同じシステムの他のジョブに影響することがあるため、System i ナビゲーター製品を使用するあらゆるユーザーにとって、ジョブのパフォーマンスは重要な問題となります。問題となりうるジョブを表示すると、パフォーマンス上の問題を未然に防ぐことができます。

「経過パフォーマンス統計」ウィンドウを使用すると、ジョブの CPU 使用、ディスク入出力 (ハード・ディスク入出力)、ページ不在率、平均応答時間、および対話式トランザクションの数をモニターできます。このウィンドウのオプションを選択して、手動でまたはスケジュール設定によりこれらの統計を最新表示することができます。

経過パフォーマンス統計を表示するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。

   注: 実行管理機能内で、ジョブを表示できる場所であればどこからでも、ジョブのパフォーマンスを表示できます。「経過パフォーマンス統計」ウィンドウは、「ジョブ」プロパティー・ウィンドウの「パフォーマンス」タブから表示できます。
2. パフォーマンス統計を表示したいジョブを右マウス・ボタンでクリックし、「**詳細**」 → 「**経過パフォーマンス統計**」をクリックします。

   パフォーマンス統計を最新表示およびリセットしたり、自動的に最新表示されるようにスケジュール設定することもできます。

   注: 複数のウィンドウを開いて、一度に複数のジョブの経過パフォーマンス統計を見ることもできます。このようにして、問題のある複数のジョブを一度に表示できます。一つのウィンドウに表示される情報は、一つのジョブに限られます。

経過パフォーマンス統計は、システム内で変動するジョブのパフォーマンスを表示できる一つの方法です。システム内のジョブを表示できる別の方法としては、「マネージメント・セントラル」フォルダーの使用なども挙げられます。マネージメント・セントラルでは、ジョブだけでなく、システム・パフォーマンスやメッセージもモニターできます。

全システム状況の表示

System i ナビゲーター では、システム状況に関連するすべての情報が一箇所にまとめられています。 これにより、システムのパフォーマンスをモニターし、問題が発生する可能性のある領域を識別し、パフォーマンスを改善するために取る必要がある処置を迅速に判別しやすくなります。

「システム状況」ウィンドウでは、全システム状況が以下の 6 つの特定の領域に分割されています。

一般 これは、CPU 経過使用量の割合、アクティブ・ジョブ数、アドレス使用率、システム・ディスク・プール使用率、システム上の合計ジョブ数、使用された永続および一時アドレスの割合、合計ディスク・スペース、およびシステム・ディスク・プール容量です。

ジョブ これは、ジョブの合計数、アクティブ・ジョブ数、ジョブの最大数、およびアクティブ・スレッド数です。

プロセッサー
これは、CPU 経過使用量の割合です。(ハードウェア構成によっては、プロセッサーのタイプ、プロセッサー数、処理能力、仮想プロセッサー、対話式パフォーマンス、経過共用プロセッサー・プール使用量、および経過制限無し CPU 容量使用量に関する追加情報が表示される場合もあります。)

メモリー
これは、システム上の合計メモリー (主記憶域) であり、システム上のアクティブ・メモリー・プールのリストにアクセスするためのボタンです。

ディスク・スペース
これは、合計ディスク・スペース、システム・ディスク・プール容量および使用量、使用された一時記憶域に関する情報であり、より詳細なディスク状況、システム情報のディスク・プールのリスト、および記憶域システム値情報にアクセスするためのボタンです。

アドレス
これは、使用された永続および一時アドレス、使用された大容量 (256 MB) 永続および一時アドレス、および使用された大容量 (4 GB) 永続および一時アドレスに関する情報です。

一般システム状況を表示するには、以下の指示に従います。

1. System i ナビゲーター で、**「ユーザー接続」**を展開します。
2. サーバーを右マウス・ボタン・クリックして、**「システム状況」**をクリックします。

「システム状況」ウィンドウが表示されます。このウィンドウについての詳細は、System i ナビゲーターのオンライン・ヘルプを参照してください。

ディスク状況の検査:

システム上のディスク装置のパフォーマンスを検査したり、それに関する状況情報を表示することが時折必要になります。

「ディスク状況 (Disk Status)」ウィンドウを表示するには、以下のステップに従います。

1. System i ナビゲーター で、**「ユーザー接続」**を展開します。
2. ***ご使用のシステム***を右クリックし、**「ディスク・スペース (Disk Space)」 → 「システム状況」**タブを選択します。
3. 「ディスク・スペース (Disk Space)」ウィンドウで、**「ディスク状況 (Disk Status)」**をクリックします。 「ディスク状況 (Disk Status)」ウィンドウが表示されます。

「ディスク状況 (Disk Status)」ウィンドウの**「この表示をカスタマイズ (Customize this view)」 → 「列 (Columns)」**オプションを使用して、以下の情報を表示します。

- 読み取り量 (KB)
- 書き込み量 (KB)
- 使用中パーセント

- 圧縮
- ディスク・プール
- 入出力要求
- 使用済みパーセント
- 保護状況
- 保護タイプ
- 読み取り要求
- 要求サイズ (KB)
- サイズ (MB)
- タイプ
- 書き込み要求

ジョブの管理

多くの実行管理機能管理者が理解しているように、ジョブを保留状態にしたり、ジョブをジョブ待ち行列から別の待ち行列に移動したりするよりもジョブを管理する機会の方が多くあります。このトピックでは、一般的なジョブ管理タスクや、システム・パフォーマンスを改善するのに役立つタスクの幾つかが取り上げられています。

一般的なジョブ・タスク

ここで取り上げるのは、ジョブで実行可能な最も一般的なタスクです。説明は、System i ナビゲーター (使用可能な場合) と文字ベース・インターフェースの両方に当てはまります。

ジョブの開始:

対話式ジョブは、ユーザーがワークステーションにサインオンすると開始します。事前開始ジョブおよびバッチ・ジョブを開始するには、環境に応じて System i ナビゲーター または文字ベース・インターフェースを使用します。

ジョブ待ち行列で待機中のバッチ・ジョブの開始:

ジョブが即時に開始するよう強制しなければならない場合があります。これを行う方法としては、使用中でないジョブ待ち行列にジョブを移動するのが最も効率的ですが、他の方法もあります。

バッチ・ジョブを開始する場合は、まずジョブが入っているジョブ待ち行列の状態を確認し、この状況でジョブを別の待ち行列に移動することに意味があるかどうかを判別してください。(**「ユーザー接続」 → 「サーバー」 → 「実行管理機能」 → 「ジョブ待ち行列」 → 「アクティブ・ジョブ待ち行列」または「すべてのジョブ待ち行列」**)

ジョブを別の待ち行列に移動するのがふさわしくない場合は、実行中のジョブを保留にしてから、開始する必要のあるジョブの優先順位を上げます。ただし、保留したジョブは最大アクティブ・ジョブ数に含まれているため、この方法を使用する場合は注意してください。

ジョブの優先順位を変更してジョブを実行する日時を指示するには、以下の指示に従います。

1. ジョブを右マウス・ボタンでクリックし、**「プロパティー」**をクリックします。
2. 「ジョブのプロパティー」ウィンドウで、**「ジョブ待ち行列」**タブをクリックします。
3. **「ジョブ待ち行列上の優先順位」**をより高い優先順位 (0 が最高) に変更します。
4. **「ジョブを実行可能にする日時」**を「今すぐ」に設定するか、日時を指定します。

5. 「**OK**」をクリックします。

事前開始ジョブの開始:

一般的な事前開始ジョブは、サブシステムの開始と同時に開始されます。エラーのためすべての事前開始ジョブがシステムによって終了された場合、または事前開始ジョブ項目の STRJOBS (*NO) のためにサブシステムの開始時にすべての事前開始ジョブが開始されなかった場合には、事前開始ジョブを手動で開始します。事前開始ジョブを開始するには、文字ベース・インターフェースを使用します。

コマンド: 事前開始ジョブの開始 (STRPJ)

STRPJ コマンドは、関連するサブシステムの開始が完了するまでは使用しないでください。必要とする事前開始ジョブが正常に開始されるようにするため、STRPJ コマンドが失敗する場合には再試行で遅延ループをコード化します。

同時にアクティブ状態にできる事前開始ジョブの数は、事前開始ジョブ項目の MAXJOBS 属性、およびサブシステムの MAXJOBS 属性によって制限されます。通信項目の MAXACT 属性は、同時に通信項目から保守できるプログラム開始要求の数を制御します。

注: STRJOBS 属性で *NO を指定した場合、サブシステムの開始時に事前開始ジョブ項目に対して事前開始ジョブは開始されません。STRPJ コマンドを実行しても、STRJOBS パラメーターの値は変更されません。

例: 次の例では、サブシステム SBS1 の事前開始ジョブ項目 PJPGM の事前開始ジョブを開始します。サブシステム SBS1 は、このコマンドの発行時にアクティブ状態になければなりません。開始されるジョブ数は、事前開始ジョブ項目 PJPGM の INLJOBS 値で指定された数です。サブシステムは、ライブラリー PJLIB 内のプログラム PJPGM を開始します。

```
STRPJ   SBS(SBS1)  PGM(PJLIB/PJPGM)
```

ジョブの終了:

ジョブを終了するには、System i ナビゲーター または文字ベース・インターフェースを使用できます。ジョブはアクティブなものでも、ジョブ待ち行列上にあるものでもかまいません。ジョブは、即時に終了することも、または時間間隔を指定してジョブ終了処理が実行されるようにすることもできます。

System i ナビゲーター:
System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーターで、「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. 終了するジョブを見つけます。
3. ジョブを右マウス・ボタンでクリックして、「**削除/終了**」をクリックします。
4. 「削除/終了の確認 (Confirm Delete/End)」ウィンドウを完成させて、「**削除**」をクリックします。

文字ベース・インターフェース:

コマンド: ジョブ終了 (ENDJOB)

終了させるジョブ待ち行列の名前が分からない場合は、ジョブ名を検索するために以下のコマンドの 1 つを使用します。

- アクティブ・ジョブ処理 (WRKACTJOB)
- ユーザー・ジョブ処理 (WRKUSRJOB)

- 投入済みジョブ処理 (WRKSBMJOB)
- サブシステム・ジョブ処理 (WRKSBSJOB)
- サブシステム終了 (ENDSBS)。このコマンドは、サブシステム内のすべてのジョブを終了させます。
- システム終了 (ENDSYS)。このコマンドは、システム上の大半のアクティビティーを終了させ、システムを制御サブシステムでコンソールだけがアクティブな状態にします。
- システム電源遮断 (PWRDWNSYS)。このコマンドは、電源遮断手順の終了および開始にシステムを備えさせます。

ジョブは、即時または制御された方法のいずれかで終了できます。ジョブは必ず制御された方法で終了することを強くお勧めします。

ジョブの終了: 制御:

制御された仕方でジョブを終了すると、ジョブで実行中のプログラムでジョブ終了終結処置を実行することができます。制御された仕方でジョブを終了するための遅延時間を指定できます。遅延時間がジョブ終了前に終わると、ジョブは即時に終了します。

ジョブ終了終結処置を実行する必要のあるアプリケーションは、ジョブが制御された仕方で終了する時期を検出する必要があります。アプリケーションがこれを検出する 3 つの方法は、以下のとおりです。

終了状況を同期的に検索する

特定の時点で、アプリケーションは実行されているジョブの終了状況を同期的に確認します。ジョブ属性検索 (RTVJOBA) CL コマンドを発行すると、ジョブの終了状況を検索できます。さらに、ジョブの終了状況を検索する幾つかの API のいずれかを使用することもできます。こうした API に関する詳細は、「*実行管理機能 ジョブ属性 (Work management job attributes)*」の体験レポートで取り上げられています。

入出力操作後にメジャー戻りコードおよびマイナー戻りコードを同期的に確認する

ディスプレイ入出力および ICF 通信入出力のどちらの場合も、メジャー戻りコード 02、またはマイナー戻りコード 09 を伴うメジャー戻りコード 03 は、ジョブが制御された仕方で終了していることを示します。

非同期シグナル SIGTERM を処理する

一部のアプリケーションはシグナル処理プログラムを使用して、ジョブの終了時のアプリケーションの終結処理を向上させます。ジョブが制御された仕方で終了中で、以下の条件すべてを満たしている場合には、システムは終了中のジョブに非同期シグナル SIGTERM を生成します。

- ジョブで、シグナルが使用可能である。
- ジョブは、SIGTERM シグナルを設定するシグナル処理プログラムである。
- ジョブは、問題フェーズで現在実行中である。

前述のいずれかの条件が満たされていないと、SIGTERM シグナルは終了中のジョブに対して生成されません。

制御された仕方で終了中のジョブに、非同期シグナル SIGTERM 用のシグナル処理プロシージャーがある場合、そのジョブに対して SIGTERM シグナルが生成されます。SIGTERM シグナル用のシグナル処理プロシージャーに制御が与えられると、そのプロシージャーは制御された仕方でアプリケーションを終了させるために適切な処置を行うことができます。

関連タスク

180 ページの『サブシステムの停止』

System i ナビゲーター または文字ベース・インターフェースを使用して、1 つまたは複数のアクティ

ブ・サブシステムを停止して、処理されているアクティブ作業をどうするか指定することができます。サブシステムが停止すると、そのサブシステムでは新しいジョブまたは経路指定ステップは開始されません。

関連情報

ジョブ・システム値: 即時終了の最大時間

ジョブの終了: 即時:

ジョブが即時に終了する場合、部分的に更新されたアプリケーション・データなどの、望ましくない結果が生じる可能性があります。即時終了オプションは、制御された終了が失敗した場合にのみ使用します。

ジョブを終了する前に、進行中の 2 フェーズ・コミット操作が原因で疑わしい状態にある作業論理単位がないことを検査する必要があります。それがあると、アクション ifENDJOB コミットメント・オプションは、ENDJOB 処理にかなり影響を与える可能性があります。このオプションは、コミットメント・オプション変更 (QTNCHGCO) API の一部です。たとえば、アクション ifENDJOB コミットメント・オプションがデフォルト値 WAIT の場合、このジョブは保留になり、コミットメント制御操作が完了するまで、ジョブ処理の終了は完了しません。これにより、関連するすべてのシステム上のデータベース保全性が保証されます。

即時終了オプションを使用する場合、システムは最小限のジョブ終了処理を実行します。これには以下を含めることができます。

- データベース・ファイルのクローズ
- 出力待ち行列へのジョブ・ログのスプール
- オペレーティング・システム内の内部オブジェクトのクリーンアップ
- ジョブ終了画面の表示 (対話式ジョブ用)
- コミットメント制御処理の完了

関連情報

Change Commitment Options (QTNCHGCO) API

ジョブの検索:

システム上のジョブを検索する方法を理解しておくことは重要です。理由はどうあれ、ある時点で特定のジョブの特定の情報が必要になる場合があります。

System i ナビゲーター で、すべてのジョブに対する「検索」を実行するか、「検索」の前に「組み込み」機能を使って検索対象を絞ることができます。 「組み込み」機能を使用すると、System i ナビゲーターに表示されるものを制限することができます。たとえば、何百というジョブに対して検索を実行する代わりに、「組み込み」を実行して、特定のタイプのジョブだけが表示されるようにできます。あるいは、特定のジョブ・ユーザー ID を持つジョブだけを表示することができます。

パフォーマンスの観点からすると、システムに数多くのジョブが存在する場合は、「組み込み」機能を使って検索対象の数を絞っておくことをお勧めします。システム上に多くのジョブが存在する場合にすべてを対象に検索をかけると、システム・パフォーマンスが低下してしまいます。

注: ジョブを探すどのような場合でも、実行管理機能全体で「検索」および「組み込み」機能を使用できます。さらに、同じ方法でこれらのツールを使用して、ジョブ待ち行列、サブシステム、およびメモリー・プールを検索することもできます。これらのツールを使用する前に、検索を行う領域をクリックする必要があることに注意してください。

System i ナビゲーター:
「**検索**」(Ctrl+F) オプションを使用してジョブを検索する場合、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. 「編集 (Edit)」メニューで、「**検索**」(Ctrl+F)を選択します。
3. 「**検索対象:**」テキスト・フィールドに、検索するジョブ ID (たとえば、Qqqtemp1) を入力します。ジョブの列はすべて検索対象です。
4. 「**検索**」をクリックします。 System i ナビゲーター では、検出されたジョブが強調表示になります。

要確認: ジョブ名は、引用符で囲む場合のみ (たとえば、″MyJob″) 大/小文字が区別されます。ジョブ名を引用符で囲まない場合は、大/小文字は区別されません。

表示される情報の制限:
表示される情報を制限するには、「組み込み」機能を使用します。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。
2. 「表示」メニューから、「**ビューのカスタマイズ (Customize this View)**」 → 「**組み込み**」を選択します。「組み込み」ウィンドウが表示されます。
3. 「組み込み」ウィンドウで、ジョブの検索に使用するオプションを選択します。
4. 「OK」をクリックします。

文字ベース・インターフェース:
システム上のジョブを検索するには、アクティブ・ジョブ処理 (WRKACTJOB)、ユーザー・ジョブ処理 (WRKUSRJOB)、または投入済みジョブ処理 (WRKSBMJOB) のいずれかのコマンドを使用します。

ジョブ待ち行列上のジョブの表示:

ジョブ待ち行列は実行管理機能で処理される作業の一部 (たとえば、一部のバッチ・ジョブ) にフィルターをかけます。ジョブ待ち行列内のジョブを表示できると、サブシステムに送信されるのを待っているのがどのジョブか表示することができます。

System i ナビゲーター:
ジョブ待ち行列上のジョブを表示するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**アクティブ・ジョブ待ち行列**」または「**すべてのジョブ待ち行列**」の順に展開します。
2. ジョブを表示したいジョブ待ち行列 (たとえば、Jobqueue1) をクリックします。そのジョブ待ち行列内のジョブが表示されます。

文字ベース・インターフェース:

コマンド: ジョブ待ち行列処理 (WRKJOBQ)

このコマンドは、システム上の使用可能なすべてのジョブ待ち行列のリストを示します。ジョブが入っているジョブ待ち行列を見付けたら、オプション「**5=処理**」を選択して、ジョブ待ち行列内のすべてのジョブを表示できます。

サブシステム・ジョブ処理コマンドを使用して、ジョブ待ち行列およびそれぞれのジョブのリストを表示することもできます。

コマンド: サブシステム・ジョブ処理 (WRKSBSJOB) SBS(*JOBQ)

サブシステム内のジョブの表示:

サブシステムはジョブの実行に使用されるワークフローとリソースを調整します。System i ナビゲーターを使用すると、サブシステム内で現在アクティブなジョブ (必ずしも実行中であるとは限らない) が何かを見ることができます。

System i ナビゲーター:
サブシステム内のジョブを表示するには、以下の手順で行います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**サブシステム**」 → 「**アクティブ・サブシステム**」の順に展開します。
2. 表示するジョブが含まれているサブシステムをクリックします。

文字ベース・インターフェース:

コマンド: アクティブ・ジョブ処理 (WRKACTJOB SBS(subsystem name)

コマンド: サブシステム記述処理(WRKSBSD)

サブシステム記述処理コマンドを使用して、サブシステムのリストを表示します。ジョブが含まれているサブシステムを見付けたら、オプション「**8=サブシステム・ジョブの処理**」を使用して、ジョブ情報を表示します。

注: ジョブ情報を表示するには、サブシステムがアクティブになっている必要があります。

ジョブ属性の表示:

ジョブ属性には、ジョブが処理される方法に関する情報が含まれています。これは、ジョブが作成されたときに指定されます。いくつかの属性は、ジョブ記述から渡されます。ジョブの作成後、ジョブ属性は System i ナビゲーターの実行管理機能によって表示および管理することができます。 System i ナビゲーター の「ジョブ・プロパティー」ページは、ジョブを管理するために効果的で使いやすい機能を提供しているため、システム・オペレーターにとって作業しやすい環境になっています。

関連情報

経験報告: 実行管理ジョブ属性

System i ナビゲーター:
ジョブ属性を表示するには、以下の指示に従います。

1. System i ナビゲーターで、処理するジョブのタイプに応じて、「**ユーザー接続**」 → 「**サーバー**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または 「**サーバー・ジョブ**」の順に展開します。
2. プロパティーを表示または変更するジョブを検索をします。
3. 「**ジョブ名**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。

ジョブ属性はどのユーザーでも表示できますが、変更できるのは適切な権限を持ったユーザーだけです。同様に、権限が与えられたユーザーはジョブ・アクションによってジョブを管理することができます。
System i ナビゲーター では、システム・ジョブの属性を変更できません。 しかし、文字ベースのインターフェースでは、システム・ジョブの変更 (CHGSYSJOB) コマンドを使用して、一部のシステム・ジョブの実行優先順位を変更できます。

文字ベース・インターフェース:

コマンド: ジョブ処理 (WRKJOB) このジョブがアクティブ状態ある場合、次の情報を表示できます。ジョブ実行属性、呼び出しスタック情報、ジョブ・ロック情報、ライブラリー・リスト情報、ジョブ・ログ情報、オープン・ファイル情報、ファイル・オーバーライド情報、コミットメント制御状況、通信状況、活動化グループ情報、相互除外情報、およびスレッド情報。

コマンド: ジョブ表示 (DSPJOB)

このコマンドは、以下のジョブに関する情報を表示します。ジョブ状況属性、ジョブ定義属性、ジョブ実行属性、スプール・ファイル情報、ジョブ・ログ情報、呼び出しスタック情報、ジョブ・ロック情報、ライブラリー・リスト情報、オープン・ファイル情報、ファイル指定変更情報、コミットメント制御状況、通信状況、活動化グループ情報、相互除外情報、スレッド情報、メディア・ライブラリー、および属性情報。

呼び出しスタックの表示:

ジョブまたはスレッドの呼び出しスタックを表示するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連概念

35 ページの『呼び出しスタック』
呼び出しスタック は、ジョブに対して現在実行しているすべてのプログラムまたはプロシージャーの番号付きリストです。プログラムおよびプロシージャーは、CALL 命令によって明示的に開始することも、他の何らかのイベントによって暗黙的に開始することもできます。

System i ナビゲーター:

1. System i ナビゲーターで、処理するジョブのタイプに応じて、「**ユーザー接続**」 → 「**サーバー**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。
2. ジョブ名を右マウス・ボタン・クリックして、「**詳細**」 → 「**呼び出しスタック**」をクリックします。

スレッドの呼び出しスタックを表示する場合は、以下の手順で行います。

1. System i ナビゲーターで、処理するジョブのタイプに応じて、「**ユーザー接続**」 → 「**サーバー**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。
2. ジョブ名を右マウス・ボタン・クリックして、「**詳細**」 → 「**スレッド**」をクリックします。
3. スレッドのリストから特定のスレッドを右マウス・ボタン・クリックして、「**詳細**」 → 「**呼び出しスタック**」をクリックします。

*SERVICE 特殊権限を持つユーザー・プロファイルの下で実行している場合で、LIC および i5/OS PASE Kernel の追加項目を表示する場合は、「呼び出しスタック」ウィンドウから、「このビューのカスタマイズ」ウィンドウの「組み込み」オプションを使用します。 (「**表示**」**メニュー** → 「**このビューのカスタマイズ**」 → 「**組み込み**」)

文字ベース・インターフェース:

コマンド: ジョブ処理 (WRKJOB) またはジョブ表示 (DSPJOB)

オプション 11 (呼び出しスタックの表示) を選択します (アクティブになっている場合)。

スレッドの呼び出しスタックを表示する場合は、WRKJOB または DSPJOB コマンドを発行した後で、オプション 20 (スレッドの処理) を選択します (アクティブになっている場合)。 次に、選択したスレッドに対してオプション 10 (呼び出しスタックの表示) オプションを選択します。

ジョブのジョブ待ち行列への配置:

既存のジョブをある待ち行列から別の待ち行列に移動するか、新規ジョブを投入することによって、ジョブはジョブ待ち行列に入れられます。待ち行列間でジョブを移動するには、System i ナビゲーター を使用します。 新規ジョブを投入するには、文字ベース・インターフェースを使用します。

System i ナビゲーター:
System i ナビゲーター インターフェースを使用するには、ジョブが別のジョブ待ち行列に存在している必要があります。 この場合、ジョブをある待ち行列から別の待ち行列に移動できます。(新規ジョブをジョブ待ち行列に入れるには、コマンド行インターフェースを使用します。)

1. System i ナビゲーターで、「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**すべてのジョブ待ち行列**」の順に展開します。
2. 移動するジョブを右マウス・ボタンでクリックします。 「Move (移動)」ウィンドウが開き、宛先待ち行列を指定できます。

文字ベース・インターフェース:
以下に、新規ジョブを新規ジョブ待ち行列に入れるための文字ベース・インターフェース・メソッドのリストを示します。

- ジョブ投入 (SBMJOB): 実行中のジョブが別のジョブをジョブ待ち行列に投入して、後からバッチ・ジョブとして実行できるようにすることができます。 要求データの 1 つの要素だけを新しいジョブのメッセージ待ち行列に入れることができます。ジョブに使用する経路指定項目が CL コマンド処理プログラム (たとえば、IBM 提供の QCMD プログラム) を指定している場合には、要求データは CL コマンドであってもかまいません。
- ジョブ・スケジュール項目追加 (ADDJOBSCDE): システムは、ジョブ・スケジュール項目で指定された日付および時刻にジョブをジョブ待ち行列に投入します。
- データベース・ジョブ投入 (SBMDBJOB): ジョブをバッチ・ジョブとして実行できるようにジョブ待ち行列に投入します。 入力ストリームは、物理データベース・ファイルから、あるいは単一レコード様式を持つ論理データベース・ファイルから読み取られます。このコマンドでは、このデータベース・ファイルとそのメンバーの名前、使用されるジョブ待ち行列の名前、および投入されるジョブが投入ジョブ処理 (WRKSBMJOB) コマンドによって表示できるかどうかを指定することができます。
- データベース読取プログラム開始 (STRDBRDR): データベースからバッチ入力ストリームを読み取り、1 つ以上のジョブをジョブ待ち行列に入れます。
- ジョブ転送 (TFRJOB): 現行ジョブをアクティブ・サブシステムの別のジョブ待ち行列に移動します。
- バッチ・ジョブ転送 (TFRBCHJOB): 現行ジョブを別のジョブ待ち行列に移動します。

別のジョブ待ち行列へのジョブの移動:

さまざまな理由により、ジョブを別の待ち行列に移動したい場合があります。たとえば、ジョブが長く実行されているため、ジョブが待ち行列内でバックログになることがあります。おそらく、ジョブのスケジュールされた実行時間が高位の優先順位を持つ新しいジョブと競合しているのかもしれません。このような状況を管理する 1 つの方法は、待機中のジョブを使用率のそれほど高くない別の待ち行列に移動することです。

ジョブを待ち行列間で移動するには、System i ナビゲーター インターフェースまたは文字ベース・インターフェースのどちらも使用できます。

System i ナビゲーター:
System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーターで、「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**すべてのジョブ待ち行列**」の順に展開します。

2. ジョブが現在含まれている待ち行列を見つけて、オープンします。
3. 移動するジョブを右マウス・ボタンでクリックします。 「移動」ウィンドウがオープンし、ターゲット待ち行列を指定できます。

   注: この待ち行列から複数のジョブを移動したい場合、各ジョブをクリックする際 CTRL キーを押したままにします。その後、右マウス・ボタン・クリックして 「**移動**」をクリックします。

- 実行を待機中のジョブは、ターゲット待ち行列上でも同じ相対位置に移動します (たとえば、ジョブ待ち行列優先順位 3 のジョブは、ターゲット待ち行列上で実行を待機中の優先順位 3 の他のジョブの後ろに移動します)。
- 保留されているジョブは、ターゲット待ち行列上でも引き続き保留され、同じ相対位置に置かれます (たとえば、ジョブ待ち行列優先順位 3 の保留ジョブは、ターゲット待ち行列上の優先順位 3 の他の保留ジョブの後ろに移動します)。
- 実行がスケジュールされているジョブは、ターゲット待ち行列に移動される際にそのスケジュールされた時間は未変更のままです。

文字ベース・インターフェース:
コマンド: ジョブ変更 (CHGJOB)

例: 次の例では、ジョブ JOBA がジョブ待ち行列 JOBQB に移動します。

```
CHGJOB JOB(JOBA) JOBQ(LIBA/JOBQB)
```

ジョブ待ち行列でのジョブの優先順位を上げる:

ジョブ待ち行列内のすべてのジョブは、処理のため列に並んで待機します。待ち行列内のあるジョブが完了すると、列上の次のジョブが開始されます。待ち行列内のジョブの処理順序は、ジョブの優先順位、およびサブシステム上で同時に実行できるジョブの最大数に依存します。

ジョブの重要性は、そのライフ・サイクルが進むにつれて変化します。他のジョブとの関係で、優先順位が高まったり、低くなったりする場合があるのです。それで、ジョブ待ち行列内でのジョブの優先順位の変更の方法を知っておく必要があります。

ジョブ待ち行列でのジョブの優先順位は、そのジョブがいつごろサブシステムに入れられて実行されるかを判別するのに役立ちます。 0 から 9 の範囲 (0 が最も重要) でジョブ待ち行列におけるジョブの優先順位を決定します。

System i ナビゲーター:
System i ナビゲーター を使用して、ジョブ待ち行列内のジョブの優先順位を変更できます。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**アクティブ・ジョブ待ち行列**」または「**すべてのジョブ待ち行列**」 → **ジョブが置かれているジョブ待ち行列**に展開します。
2. ジョブを右マウス・ボタンでクリックし、「**プロパティー**」をクリックします。
3. 「Job - Properties (ジョブ - プロパティー)」ウィンドウで、「**ジョブ待ち行列**」タブをクリックします。
4. 「**ジョブ待ち行列上の優先順位**」リストから、今よりも高い (または低い) 優先順位番号を選択します。ジョブ待ち行列の優先順位は 0 から 9 であり、0 が最高の優先順位です。
5. 「**OK**」をクリックします。ジョブのジョブ待ち行列優先順位が変更されます。たとえば、優先順位 4 のジョブを優先順位 3 に変更すると、優先順位 3 のジョブのリストの一番下に移動します。
6. F5 を押して、「ジョブ待ち行列」ウィンドウを最新表示します。

文字ベース・インターフェース:

コマンド: ジョブ変更 (CHGJOB)

パラメーター: JOBPTY

例: このコマンドは、ジョブ PAYROLL のスケジューリング優先順位を 4 に変更します。簡単な名前のジョブを 1 つだけ指定したので、システムには PAYROLL という名前のジョブしかありません。複数のジョブが存在する場合、デフォルトの DUPJOBOPT(*SELECT) により、対話式ジョブの選択パネルが表示されます。

```
CHGJOB   JOB(PAYROLL)  JOBPTY(4)
```

ジョブ優先順位の設定に関するヒント:

バッチ環境で実行するジョブの優先順位は、通常、対話環境のジョブの優先順位よりも低くしてください。さらに、ループ・プログラムがプロセッサー時間およびアクティビティー・レベルを占有することがないように、タイム・スライスを短くします。

システム・オペレーターのジョブの優先順位を他のジョブの優先順位よりも高くして、システム・オペレーターがシステムに対する必要な応答を効果的に行いたいという場合もあります。

制御サブシステムとして QCTL を使用する場合、オペレーターはコンソールにサインオンした後には自動的に高位の優先順位で実行します。なぜなら、QCTL は、高位の優先順位を指定する QCTL クラスを使用してコンソール・ジョブを経路指定するからです。

オペレーターが高位の優先順位で実行できるようにシステムをセットアップする別の方法は、以下のとおりです。

1. 固有の経路指定データを伴う経路指定項目をサブシステムに追加して、QSYS/QCTL クラスを指定します。
2. オペレーターに、経路指定項目で使用したのと同じ固有の経路指定データを指定した新しいジョブ記述を作成します。
3. オペレーターのユーザー・プロファイルを変更して、新しいジョブ記述を指定します。
4. オペレーターがそのサブシステムにサインオンすると、ジョブは QCTL クラスを使用して経路指定し、通常の対話式ジョブで使用されるクラスよりも高位の優先順位を指定します。

このジョブ実行優先順位は、ジョブ内で実行されるスレッドで最高の優先順位になります。各スレッドは、このジョブ優先順位よりも低い独自のスレッド優先順位を有することになります。ジョブ変更 (CHGJOB) コマンドは、ジョブ優先順位のみを変更します。 ジョブ変更 (QWTCHGJB) API は、ジョブ優先順位またはスレッド優先順位のどちらも変更できます。

ジョブの 1 回の投入:

即時にまたはスケジュールした日時にジョブを 1 回実行する必要がある場合は、ジョブ投入 (SBMJOB) コマンドを使用します。 この方法では、ジョブはジョブ待ち行列に即時に入れられます。

バッチ・ジョブを 1 回投入するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ投入 (SBMJOB)

SBMJOB コマンドは、ジョブ記述、CL コマンドまたは要求データ、またはプログラムを実行する経路指定データを指定することによって、ジョブをバッチ・ジョブ待ち行列に投入します。単一の CL コマンドをバッチ・ジョブで実行する場合は、SBMJOB で CMD パラメーターを使用します。これにより、構文検査が行われ、プロンプトが出されます。

例: 以下の例では、SBMJOB コマンドは、ジョブ記述 QBATCH を使用して WSYS という名前のジョブをジョブ待ち行列 QBATCH に投入します。 CMD パラメーターは、ジョブ内で実行される CL コマンドを指定します。

```
SBMJOB JOBD(QBATCH) JOB(WSYS) JOBQ(QBATCH) CMD(WRKSYSSTS)
```

関連概念

66 ページの『ジョブ投入コマンド』
この文字ベース・インターフェース・コマンドは、ジョブ待ち行列内でジョブが解放される時を制御します。これは、一度だけしか実行する必要がないジョブをスケジュールするための簡単な方法です。これにより、現行のジョブに対して定義されている多くのジョブ属性を使用できます。

ジョブ類縁性情報の表示:

システム上のジョブは、それぞれがメモリーおよびプロセッサーの類縁性情報を持っています。

この類縁性情報には、複数のスレッドが開始されたときに、それらが初期スレッドと同じプロセッサーやメモリーに対して類縁性を持つかどうかが記述されています。 また、スレッドとスレッドが割り当てられたシステムのサブセットの間で、システムが類縁性を維持しようとする度合いも指定されています。さらに、類縁性情報には、あるジョブが他のジョブと同じグループに属しているかどうかも明示されており、同じグループに属するジョブが同一のシステム・リソースのサブセットに類縁性を持てるようにもなっています。

主記憶域のデータ・セットの一部を共用するスレッドをグループ化することによって、システムのキャッシング速度およびメモリー・アクセス速度が向上します。

System i ナビゲーター:

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。

   注: 実行管理機能内で、ジョブを表示できる場所であればどこからでも、ジョブの類縁性情報を表示できます。
2. 表示するジョブを右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。
3. 「リソース (Resources)」ページで、「**メモリーおよびプロセッサーの類縁性 (Memory and processor affinity)**」情報を表示できます。

文字ベース・インターフェース:

コマンド: ジョブ処理 (WRKJOB)

オプション 3:「Display job run attributes, if active (ジョブ実行属性がアクティブであれば表示)」を選択します。

ジョブ記述の管理

ジョブ記述は特定のジョブ関連属性の集合を収集したものなので、同じジョブ記述を複数のジョブで使用できます。ですから、ジョブ記述を使用すると、各ジョブに対して同じパラメーターを繰り返して指定する必

要はありません。バッチ・ジョブまたは対話式ジョブについて記述するためにジョブ記述を作成できます。さらに、システムの各ユーザー用に固有の記述を作成できます。ジョブ記述は、文字ベース・インターフェースを使用して、作成および管理を行えます。

ジョブ記述の作成:

文字ベース・インターフェース、ジョブ記述処理 (WRKJOBD) コマンド、またはジョブ記述作成 (CRTJOBD) コマンドを使用して、ジョブ記述を作成します。

コマンド: ジョブ記述作成 (CRTJOBD)

例: この例では、ジョブ記述はユーザーの現行ライブラリー内に INT4 という名前で作成されます。このジョブ記述は対話式ジョブ用であり、Department 127 により使用されます。サインオン時に、パスワードを入力する必要があります。文字 QCMDI は、ジョブが実行されるサブシステムの経路指定テーブルと比較される経路指定データとして使用されます。すべての照会メッセージはシステム応答リスト内の項目と比較され、応答を自動的に発行するかどうかが決定されます。

```
CRTJOBD   JOBD(INT4)  USER(*RQD)  RTGDTA(QCMDI)
          INQMSGRPY(*SYSRPYL)
          TEXT('Interactive #4 JOBD for Department 127')
```

このコマンドは、ユーザーの現行ライブラリー内に BATCH3 という名前のジョブ記述を作成します。この記述を使用するジョブは、ジョブ待ち行列 NIGHTQ に置かれます。この記述およびそのスプール出力を使用するジョブの優先順位は 4 です。 QCMDB は、ジョブが実行されるサブシステムの経路指定テーブル内の項目と比較される経路指定データです。アカウンティング・コード NIGHTQ012345 は、このジョブ記述を使用するジョブのアカウンティング統計の記録時に使用されます。

```
CRTJOBD   JOBD(BATCH3)  USER(*RQD)  JOBQ(NIGHTQ)  JOBPTY(4)
          OUTPTY(4) ACGCDE(NIGHTQ012345)  RTGDTA(QCMDB)
          TEXT('Batch #3 JOBD for high priority night work')
```

注: ジョブ記述内の値は、一般にはバッチ・ジョブ (BCHJOB) およびジョブ投入 (SBMJOB) コマンドで、そのパラメーターが指定されない場合に、対応するパラメーターのデフォルト値として使用されます。ジョブ記述内の値は、BCHJOB および SBMJOB コマンドに指定された値により指定変更できます。

関連概念

33 ページの『ジョブ記述』
ジョブ記述により、複数のユーザー用に保管されて使用できる、ジョブ属性のセットを作成できます。ジョブ記述は、いくつかのジョブ属性のソースとして使用でき、システムにジョブの実行方法を通知します。属性情報は、ジョブの開始時刻、ジョブの送信元、ジョブの実行方法などの情報をシステムに通知します。ジョブ記述は多くのジョブが使用可能なテンプレートと見なすことができ、それによって各ジョブに設定する必要のある特定のパラメーターの数を減らすことができます。

ジョブ記述の変更:

文字ベース・インターフェース、ジョブ記述処理 (WRKJOBD) コマンド、またはジョブ記述変更 (CHGJOBD) コマンドを使用して、ジョブ記述を変更することができます。

コマンド: ジョブ記述変更 (CHGJOBD)

ジョブ記述の変更後に開始され、そのジョブ記述を使用するすべてのジョブが影響を受けます。ジョブ記述で指定されているもの以外のジョブ・パラメーターに変更した場合、そのパラメーターは影響を受けません。

ジョブ記述の使用:

ジョブ記述の最も一般的な使用法は、ジョブ投入 (SBMJOB) コマンドに指定するという方法です。ジョブ記述 (JOBD) パラメーターに、このジョブで使用するジョブ記述を指定します。 バッチ・ジョブを指定する場合は、以下の 2 つの方法のいずれかによりジョブ記述を使用できます。

- 属性を指定変更することなく指定のジョブ記述を使用する。例:

```
SBMJOB JOB(OEDAILY) JOBD(QBATCH)
```

- 属性の一部をオーバーライドして指定のジョブ記述を使用する (BCHJOB または SBMJOB コマンドを使用)。 例えば、ジョブ記述 QBATCH のメッセージ・ロギングを指定変更するには、以下のように指定します。

```
SBMJOB JOB(OEDAILY) JOBD(QBATCH) LOG(2 20 *SECLVL)
```

ジョブ記述パラメーターをサポートする追加コマンドを以下に示します。

- バッチ・ジョブ (BCHJOB): このコマンドは、バッチ入力ストリーム内のバッチ・ジョブの開始を指示します。また、このジョブの属性に対して、このジョブのジョブ記述またはユーザー・プロファイルに指定された属性値とは異なる値を指定することもできます。BCHJOB コマンドでコーディングされていない大部分のパラメーターには、ジョブ記述またはそのジョブ記述に指定されたユーザー・プロファイルに含まれている値が使用されます。
- 事前開始ジョブ項目の追加 (ADDPJE): 事前開始ジョブ項目の追加 (ADDPJE) コマンドは、指定されたサブシステム記述に事前開始ジョブ項目を追加します。 この項目は、サブシステムの開始時または事前開始ジョブの開始 (STRPJ) コマンドの入力時に開始できる事前開始ジョブを識別します。
- 自動開始ジョブ項目追加 (ADDAJE): 自動開始ジョブ項目追加 (ADDAJE) コマンドは、指定されたサブシステム記述に自動開始ジョブ項目を追加します。この項目は、ジョブを自動開始するために使用されるジョブ名およびジョブ記述を識別します。
- ワークステーション項目追加 (ADDWSE): ワークステーション項目追加 (ADDWSE) コマンドは、指定されたサブシステム記述にワークステーション項目を追加します。 各項目には、サブシステムによって制御される 1 つまたは複数のワークステーションが記述されています。ワークステーション項目で識別されるワークステーションは、サブシステムにサインオンして (または入り)、ジョブを実行できます。

注: 自動開始ジョブ、ワークステーション・ジョブ、または通信ジョブのジョブ記述属性を指定変更することはできません。

関連概念

33 ページの『ジョブ記述』
ジョブ記述により、複数のユーザー用に保管されて使用できる、ジョブ属性のセットを作成できます。ジョブ記述は、いくつかのジョブ属性のソースとして使用でき、システムにジョブの実行方法を通知します。属性情報は、ジョブの開始時刻、ジョブの送信元、ジョブの実行方法などの情報をシステムに通知します。ジョブ記述は多くのジョブが使用可能なテンプレートと見なすことができ、それによって各ジョブに設定する必要のある特定のパラメーターの数を減らすことができます。

ジョブ属性ソースの制御:

サブシステムがジョブに割り当てる属性は、ジョブ記述、ユーザーのユーザー・プロファイル、システム値、ジョブ投入 (SBMJOB) コマンドを発行するジョブ、およびワークステーション (対話式ジョブのみ) の 5 つソースに由来します。 ジョブ記述でソースを指定して、特定のジョブ属性をサブシステムが検索する場所を制御します。 ジョブ記述を変更するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ記述変更 (CHGJOBD)

ジョブ属性を制御し、異なるシステム・オブジェクトからジョブ属性を取得する場所およびタイミングをサブシステムに指示するには、以下のいずれかの方法を使用します。

- *JOBD: ジョブ記述からジョブ属性を取得するようジョブに指示します。
- *USRPRF: ユーザーのユーザー・プロファイルからジョブ属性を取得するようジョブに指示します。
- *SYSVAL: システム値からジョブ属性を取得するようジョブに指示します。
- *CURRENT: ジョブ投入 (SBMJOB) コマンドを発行するジョブからジョブ属性を取得するようジョブに指示します。
- *WRKSTN: ジョブ (対話式ジョブのみ) を使用するワークステーションからジョブ属性を取得するようジョブに指示します。

ジョブ記述の削除:

文字ベース・インターフェース、ジョブ記述処理 (WRKJOBD) コマンド、またはジョブ記述削除 (DLTJOBD) コマンドを使用して、ジョブ記述を削除します。

コマンド: ジョブ記述削除 (DLTJOBD)

注: 進行中のジョブはこのコマンドの影響を受けません。

バッチ・ジョブの管理

実行にユーザー対話を必要としないジョブは、バッチ・ジョブとして処理できます。一般にバッチ・ジョブは優先順位の低いジョブであり、実行するには特別なシステム環境が必要です。

バッチ・ジョブの投入:

一般に、バッチ・ジョブは、実行に特殊なシステム環境を必要とする優先順位の低いジョブであるため (夜間に実行するなど)、バッチ・ジョブ待ち行列に入れられます。ジョブ待ち行列内で、バッチ・ジョブはランタイム・スケジュールおよび優先順位を受け取ります。ジョブをバッチ・ジョブ待ち行列に投入するには、文字ベース・インターフェースおよび以下の 2 つのコマンドのいずれかを使用します。

コマンド: ジョブ投入 (SBMJOB)

コマンド: データベース・ジョブ投入 (SBMDBJOB)

これらのコマンドの相違は、ジョブのソースにあります。

- SBMJOB コマンドは、ジョブ記述、CL コマンドまたは要求データ、またはプログラムを実行する経路指定データを指定することによって、ジョブをバッチ・ジョブ待ち行列に投入します。単一の CL コマンドをバッチ・ジョブで実行する場合は、SBMJOB で CMD パラメーターを使用します。これにより、構文検査が行われ、プロンプトが出されます。
- SBMDBJOB コマンドは、データベース・ファイルからジョブをバッチ・ジョブ待ち行列に投入する場合に使用します。 これらのジョブの場合、入力ストリームの BCHJOB ステートメントのジョブ記述が使用されます。

例: 以下の例では、SBMJOB コマンドは、ジョブ記述 QBATCH を使用して WSYS という名前のジョブをジョブ待ち行列 QBATCH に投入します。 CMD パラメーターは、ジョブ内で実行される CL コマンドを指定します。

```
SBMJOB JOBD(QBATCH) JOB(WSYS) JOBQ(QBATCH) CMD(WRKSYSSTS)
```

注: ジョブが投入されなかったというメッセージが出される場合は、ジョブ・ログ・スプール・ファイルを表示してエラーを見付けることができます。WRKJOB コマンドを使用します。スケジュールされなかったジョブを指定し、オプション 4 (スプール・ファイル) を選択します。ジョブ・ログ・スプール・ファイルを表示して、エラーを見付けます。

関連概念

43 ページの『バッチ・ジョブの開始方法』
ユーザーがバッチ・ジョブを投入すると、ジョブ待ち行列に入れられる前に、ジョブはいくつかのシステム・オブジェクトから情報を収集します。

66 ページの『ジョブ投入コマンド』
この文字ベース・インターフェース・コマンドは、ジョブ待ち行列内でジョブが解放される時を制御します。これは、一度だけしか実行する必要がないジョブをスケジュールするための簡単な方法です。これにより、現行のジョブに対して定義されている多くのジョブ属性を使用できます。

関連情報

QPRTJOB ジョブ

インライン・データ・ファイル:

インライン・データ・ファイルは、読み取りプログラムまたはジョブ投入依頼コマンドによってジョブが読み取られるとき、バッチ・ジョブの一部として含まれるデータ・ファイルです。SBMDBJOB または STRDBRDR を使用して、 CL バッチ・ストリーム (実行する CL コマンドのストリーム) に待ち行列を作成します。その CL バッチ・ストリームには、インライン・データ・ファイル (一時ファイル) に配置するデータを含めることができます。ジョブが終了すると、インライン・データ・ファイルは削除されます。

インライン・データ・ファイルは、ジョブの中で、ファイルの開始を //DATA コマンドにより、ファイルの終了をデータ終了区切り文字によって、それぞれ区切られます。

データ終了区切り文字は、ユーザー定義の文字ストリングでもデフォルトの // でもかまいません。 // は 1 桁目および 2 桁目にも存在しなければなりません。データの 1 桁目および 2 桁目に // を含んでいる場合は、// *** END OF DATA のような固有の文字のセットを使用しなければなりません。これを固有のデータ終了区切り文字として指定するには、//DATA コマンドの ENDCHAR パラメーターを次のようにコーディングしなければなりません。

```
ENDCHAR('// *** END OF DATA')
```

注: インライン・データ・ファイルにアクセスすることができるのは、バッチ・ジョブの最初の経路指定ステップ中だけです。バッチ・ジョブにジョブ転送 (TFRJOB) コマンド、ジョブ経路再指定 (RRTJOB) コマンド、またはバッチ・ジョブ転送 (TFRBCHJOB) コマンドが入っている場合は、新しい経路指定ステップでインライン・データ・ファイルにアクセスすることはできません。

インライン・データ・ファイルには、名前が付いていてもいなくても問題ありません。名前のないインライン・データ・ファイルの場合は、QINLINE が //DATA コマンドの中でファイル名として指定されるか、名前は指定されないかどちらかです。名前付きインライン・データ・ファイルの場合、ファイル名が指定されます。

名前付きインライン・データ・ファイル には以下の特徴があります。

- ジョブの中で固有の名前を持ちます。他のインライン・データ・ファイルが同じ名前を持つことはありません。
- ジョブの中で複数回使用できます。
- オープンのたびに最初のレコードに位置合わせされます。

名前付きインライン・データ・ファイルを使用するには、プログラムでファイル名を指定するか、プログラムで指定されているファイル名を指定変更コマンドを使用してインライン・データ・ファイルの名前に変更する必要があります。ファイルは、入力専用としてオープンされます。

名前のないインライン・データ・ファイル には以下の特性があります。

- 名前は QINLINE です。(1 つのバッチ・ジョブの中では、名前のないインライン・データ・ファイルには、すべて同一名が与えられます。)
- ジョブの中で使用できるのは 1 回だけです。
- 1 つのジョブの中に名前のないインライン・データ・ファイルが複数個含まれるときは、それらのファイルはファイルのオープン時と同じ順序で入力ストリームの中になければなりません。

名前のないインライン・データ・ファイルを使用するには、次のいずれかのようにします。

- プログラムで QINLINE を指定します。
- 指定変更ファイル・コマンドを使用して、プログラムの中で指定されているファイル名を QINLINE に変更します。

使用している高水準言語が 1 つのプログラム内で固有のファイル名を必要とする場合は、QINLINE をファイル名として使用できるのは 1 回だけです。名前のないインライン・データ・ファイルを複数個使用する必要がある場合は、指定変更ファイル・コマンドをプログラムの中で使用して、その他の名前のないインライン・データ・ファイルに対して QINLINE を指定することができます。

注: コマンドを条件付きで実行し、名前のないインライン・データ・ファイルを複数処理する場合、名前のない間違ったインライン・データ・ファイルを使用すると、結果は予測できません。

インライン・データ・ファイルを開く際の考慮事項:

インライン・データ・ファイルを開くときはこれらの要素を検討する必要があります。

- レコード長は、入力レコードの長さを指定します。 (レコード長はオプションです。) レコード長がデータの長さを超える場合、メッセージがプログラムに送信されます。データにはブランクが埋め込まれます。レコード長がデータ長より短い場合、レコードは切り捨てられます。
- ファイルがプログラムで指定されると、システムはそのファイルを、ライブラリー内から検索する前に、名前付きインライン・データ・ファイルとして検索します。 したがって、名前付きインライン・データ・ファイルが、インライン・データ・ファイルではないファイルと同じ名前を持っている場合、ファイル名がライブラリー名で修飾されていないとしても、必ずそのインライン・データ・ファイルが使用されます。
- 名前付きインライン・データ・ファイルは、ファイル作成コマンドまたはファイル指定変更コマンドで SHARE(*YES) を指定すると、同じジョブ内のプログラム間で共有できます。たとえば、INPUT という名前のファイルと SHARE(*YES) を指定するファイル指定変更コマンドが、INPUT という名前のインライン・データ・ファイルを持つバッチ・ジョブ内にある場合、ファイル名 INPUT を指定している、そのジョブ内で実行しているプログラムは、その同じ名前のインライン・データ・ファイルを共有します。名前がないインライン・データ・ファイルを同じジョブ内のプログラム間で共有することはできません。
- インライン・データ・ファイルを使用する場合、//DATA コマンドに正しいファイル・タイプを指定していることを必ず確認してください。 たとえば、ファイルがソース・ファイルとして使用される場合、//DATA コマンドに指定するファイル・タイプはソースでなければなりません。
- インライン・データ・ファイルは入力に対してのみ開く必要があります。

ジョブ待ち行列で待機中のバッチ・ジョブの開始:

ジョブが即時に開始するよう強制しなければならない場合があります。これを行う方法としては、使用中でないジョブ待ち行列にジョブを移動するのが最も効率的ですが、他の方法もあります。

バッチ・ジョブを開始する場合は、まずジョブが入っているジョブ待ち行列の状態を確認し、この状況でジョブを別の待ち行列に移動することに意味があるかどうかを判別してください。(**「ユーザー接続」 → 「サーバー」 → 「実行管理機能」 → 「ジョブ待ち行列」 → 「アクティブ・ジョブ待ち行列」または「すべてのジョブ待ち行列」**)

ジョブを別の待ち行列に移動するのがふさわしくない場合は、実行中のジョブを保留にしてから、開始する必要のあるジョブの優先順位を上げます。ただし、保留したジョブは最大アクティブ・ジョブ数に含まれているため、この方法を使用する場合は注意してください。

ジョブの優先順位を変更してジョブを実行する日時を指示するには、以下の指示に従います。

1. ジョブを右マウス・ボタンでクリックし、**「プロパティー」**をクリックします。
2. 「ジョブのプロパティー」ウィンドウで、**「ジョブ待ち行列」**タブをクリックします。
3. **「ジョブ待ち行列上の優先順位」**をより高い優先順位 (0 が最高) に変更します。
4. **「ジョブを実行可能にする日時」**を「今すぐ」に設定するか、日時を指定します。
5. **「OK」**をクリックします。

関連概念

43 ページの『バッチ・ジョブの開始方法』
ユーザーがバッチ・ジョブを投入すると、ジョブ待ち行列に入れられる前に、ジョブはいくつかのシステム・オブジェクトから情報を収集します。

関連情報

QPRTJOB ジョブ

対話式ジョブの管理

対話式ジョブは、システムにサインオンする際、または 2 次またはグループ・ジョブに転送すると開始されます。対話式ジョブは、サインオフすると終了します。ディスプレイ装置から作業する場合、システムと対話するにはコマンドを発行したり、ファンクション・キーを使用したり、プログラムおよびアプリケーションを実行します。以下の情報では、対話式ジョブを管理して制御するための様々な方法が取り上げられています。

非アクティブ・ジョブおよびワークステーションの制御:

非アクティブ・ジョブ・タイムアウト間隔 (QINACTITV) システム値で時間間隔を指定すると、サブシステムがメッセージを送信するまでの間 (タイムアウトとも呼ばれる) にワークステーションが非アクティブ状態であり続けることの可能な時間を制御できます。非アクティブ・ジョブの制御によりセキュリティーが提供され、ユーザーがサインオンした画面を非アクティブ状態にしたまま離れることがなくなります。

ワークステーションが非アクティブ状態にあることをシステムが判別する方法

以下の事柄がすべて真の場合、サブシステムはワークステーションが非アクティブ状態にあると判別します。

- タイマー間隔時に、ジョブはトランザクションを何も処理していない。

  注: トランザクションとは、スクロール、Enter の押し下げ、ファンクション・キーの押し下げなど、任意のオペレーター対話と定義されます。ワークステーションにおける Enter を押さないタイプ入力は、トランザクションとは見なされません。ワークステーションでのジョブが非アクティブ基準を満たさない場合、ジョブはアクティブ状態にあると見なされます。
- ジョブ状況が表示待ちである。
- ジョブは切断されていない。

- ジョブ状況に変更がない。
- ジョブを実行中のサブシステムが制限状態にない。

非アクティブ・ジョブの処理

システムで検出される非アクティブ・ジョブを処理するには、ジョブがタイムアウトになったとき (QINACTMSGQ) システム値を使用します。以下から選択して、処理オプションを決定してください。

- QINACTMSGQ システム値にメッセージ待ち行列名を設定します。

  QINACTMSGQ システム値にメッセージ待ち行列名を指定すると、ユーザーまたはプログラムはそのメッセージ待ち行列をモニターし、ジョブの終了などの必要なアクションを実行できます。

  一対の 2 次ジョブのあるワークステーションが非アクティブ状態にある場合には、システムは 2 つのメッセージ (対の 2 次ジョブごとに 1 つずつ) をメッセージ待ち行列に送信します。その後、ユーザーまたはプログラムは ENDJOB コマンドを両方の 2 次ジョブまたはその片方に使用するか、DSCJOB コマンドを画面のアクティブ・ジョブに使用できます。
- QINACTMSGQ システム値に *DSCJOB を設定します。

  QINACTMSGQ システム値に *DSCJOB を設定すると、システムはワークステーションのすべてのジョブを切断します。システムは、ワークステーション上のすべてのジョブが QSYSOPR から、または構成済みメッセージ待ち行列から切断されたことを示すメッセージを送信します。(構成済みメッセージ待ち行列は、ディスプレイ装置記述の MSGQ パラメーターで指定されたメッセージ待ち行列です。デフォルトでは、QSYS または QSYSOPR です。)対話式ジョブがジョブの切断をサポートしていない場合 (たとえば、QPADEVxxxx 装置記述を使用する TELNET セッション)、ジョブは切断ではなく終了します。

  ジョブが非アクティブ状態にある各間隔では、メッセージは送信され続けます。
- システム値 QINACTMSGQ を *ENDJOB に設定します。

  QINACTMSGQ システム値に *ENDJOB を設定すると、システムはワークステーションのすべてのジョブを終了します。システムは、ワークステーション上のすべてのジョブが QSYSOPR に対して、または構成済みメッセージ待ち行列に対して終了したことを示すメッセージを送信します。

  注: ソース・パススルー・ジョブ、クライアント VTM (仮想端末管理機能) ジョブ、および 3270 装置エミュレーション・ジョブは、常に非アクティブ状態ですのでタイムアウトから除外されます。System/36 環境 MRT ジョブも、バッチ・ジョブとして現れるため除外されます。

対話式ジョブの終了:

対話式ジョブを終了するには、いくつかの異なる方法を使うことができます。

System i ナビゲーター を使用してジョブを終了できます。

1. 「削除/終了の確認 (Confirm Delete/End)」ウィンドウから、対話式ジョブを制御された方法または即時のいずれかで終了させることを指定できます。
2. ジョブ終了 (ENDJOB) 文字ベース・インターフェース・コマンドを使用できます。
3. 文字ベース・インターフェースを使用して対話式ジョブを即時に終了させるには、ワークステーションでサインオフ (SIGNOFF) コマンドを使用します。 ネットワークを介して接続を終了させるには、SIGNOFF コマンド上で接続終了パラメーター (ENDCNN) を使用します。
4. 装置からすべてのジョブを切断するには、ジョブ切断 (DSCJOB) コマンドを使用します。

System i ナビゲーターおよび「削除/終了の確認 (Confirm Delete/End)」ウィンドウを使用するには、以下の指示に従います。

1. **「ユーザー接続」 → 「エンドポイント・システム (End point system)」 → 「実行管理機能」 → 「アクティブ・ジョブ」**の順に展開します。
2. 終了したいジョブを右マウス・ボタンでクリックして、**「削除/終了」**をクリックします。 「削除/終了の確認 (Confirm Delete/End)」ウィンドウが表示され、対話式ジョブを終了させる方法および時を指定できます。

注: ワークステーションに関連付けられたすべての対話式ジョブを終了させるか、または (ジョブがグループ・ジョブの場合) グループに関連付けられたすべてのジョブを終了させるには、**「関連対話式ジョブのアクション (Action for related interactive jobs)」**フィールドの値を、「グループ・ジョブの終了 (End for group jobs)」または「すべての終了 (End all)」(ENDJOB コマンドの ADLINTJOBS パラメーターと同等) のいずれかに設定します。

指定された期間の間に対話式ジョブが非アクティブである場合、サブシステムに要求して、メッセージをメッセージ待ち行列に送信することもできます。次いでユーザー、またはメッセージ待ち行列をモニターしているプログラムは、ジョブを終了または切断できます。

関連概念

47 ページの『対話式ジョブの切断』
ジョブ切断 (DSCJOB) コマンドを呼び出すと、ジョブは切断され、サインオン画面が再表示されます。ジョブに再接続するには、切断を実行した同じ装置にサインオンします。別の対話式ジョブが、異なるユーザー名の下の装置上で開始される場合があります。

装置からのすべてのジョブの切断:

ジョブ切断 (DSCJOB) コマンドにより、対話式ユーザーはワークステーションですべての対話式ジョブを切断して、サインオン画面に戻ることができます。 交換回線は、それがこのワークステーションのワークステーション装置記述で指定されており、その回線上の他のワークステーションはアクティブでない場合にのみドロップできます。切断ジョブのタイムアウト間隔 (QDSCJOBITV) システム値の切断間隔に達したときにジョブが切断されると、ジョブは終了し、ジョブ・ログはジョブのスプール出力には含まれません。

制限事項:

1. 切断されるジョブは対話式ジョブでなければなりません。
2. 保留中のジョブは切断できません。
3. パススルー・ジョブは、ユーザーがシステム要求機能を使用して、パススルー・ターゲット・システムからソース・システムに戻るまでは切断できません。
4. コマンドは切断されるジョブ内から発行されるか、またはコマンドの発行者は切断されるジョブのジョブ・ユーザー ID と同じユーザー・プロファイル下で実行しているか、またはコマンドの発行者はジョブ制御 (*JOBCTL) 特殊権限を持つユーザー・プロファイル下で実行している必要があります。
5. ジョブ・ユーザー ID は、ジョブが他のジョブに認識されるために使用するユーザー・プロファイルの名前です。
6. PC オーガナイザーがアクティブの場合、ジョブは切断できません。

コマンド: ジョブ切断 (DSCJOB)

関連概念

47 ページの『対話式ジョブの切断』
ジョブ切断 (DSCJOB) コマンドを呼び出すと、ジョブは切断され、サインオン画面が再表示されます。

ジョブに再接続するには、切断を実行した同じ装置にサインオンします。別の対話式ジョブが、異なるユーザー名の下の装置上で開始される場合があります。

ジョブ切断に関する考慮事項:

ジョブを切断する場合には必ず考慮する必要のある事柄がいくつかあります。

- 「システム要求」メニューにあるオプションを使用すると対話式ジョブを切断でき、サインオン画面が表示されます。 このオプションは、ジョブの切り離し DSCJOB コマンドを呼び出します。
- ジョブを再び接続すると、プログラム、メニュー、および現行ライブラリー用にサインオン画面で指定した値は無視されます。
- PC オーガナイザーまたは PC テキスト援助機能がアクティブになっているジョブを切断することはできません。
- TCP/IP TELNET ジョブは、セッションが**ユーザー指定の名前付き**装置記述を使用している場合、切断可能です。以下のいずれかの方法で、ユーザー指定名の装置記述を作成できます。
  - DISPLAY NAME パラメーターを指定したネットワークステーションを使用する
  - System i Client Access サポートをワークステーション ID 機能と共に使用する
  - TCP/IP TELNET 装置初期化出口点を使用してワークステーション名を指定する
- 何らかの理由でジョブを切断できない場合、そのジョブは終了します。
- サブシステム終了時には、サブシステムのうち切断されているすべてのジョブが終了します。サブシステムの終了中には、サブシステム内のどのジョブにおいても DSCJOB コマンドを発行することはできません。
- 切断ジョブ間隔 (QDSCJOBITV) システム値を使用して、ジョブを切断できる時間間隔を指示することができます。この時間間隔に達すると、切断されたジョブは終了します。
- QDSCJOBITV システム値を超えていない切断ジョブは、サブシステム終了時または IPL の実行時に終了します。

関連概念

47 ページの『対話式ジョブの切断』
ジョブ切断 (DSCJOB) コマンドを呼び出すと、ジョブは切断され、サインオン画面が再表示されます。ジョブに再接続するには、切断を実行した同じ装置にサインオンします。別の対話式ジョブが、異なるユーザー名の下の装置上で開始される場合があります。

ワークステーションからの長時間実行機能の回避:

ワークステーションからの長時間実行機能 (保管/復元など) をタイアップなしで避けるために、システム・オペレーターはジョブをジョブ待ち行列に投入できます。

IBM 提供のサブシステム記述 QSYS/QBATCH または QSYS/QBASE には、この目的で使用できるジョブ待ち行列 QSYS/QBATCH があります。 固有のサブシステムを作成した場合は、そのサブシステム用のジョブ待ち行列を参照する必要があります。システム・オペレーターは、コマンドをシステム・オペレーター・メニューから投入できます。

以下は、長時間実行コマンドの投入の例です。

```
SBMJOB JOB(SAVELIBX) JOBD(QBATCH) JOBQ(QSYS/QBATCH)
       CMD(SAVLIB LIBX DEV(DKT01))
```

関連概念

46 ページの『対話式ジョブの開始方法』
ユーザーがシステムにサインオンすると、対話式ジョブが実行可能になる前に、サブシステムはいくつかのシステム・オブジェクトから情報を収集します。

事前開始ジョブの管理

プログラム開始要求を処理するのに必要な時間を減らすために、事前開始ジョブを使用することができます。これらは、実行可能な事前開始ジョブに関連した最も一般的なタスクです。

関連概念

101 ページの『事前開始通信ジョブおよびジョブ会計』
システムでジョブ会計を使用する場合、プログラム開始要求が事前開始ジョブに添付された直後に、事前開始ジョブの変更 (CHGPJ) コマンドを会計コード・パラメーター用のプログラム開始要求値 (CHGPJ ACGCDE(*PGMSTRRQS)) を使用して実行する必要があります。

事前開始ジョブの開始:

一般的な事前開始ジョブは、サブシステムの開始と同時に開始されます。エラーのためすべての事前開始ジョブがシステムによって終了された場合、または事前開始ジョブ項目の STRJOBS (*NO) のためにサブシステムの開始時にすべての事前開始ジョブが開始されなかった場合には、事前開始ジョブを手動で開始します。事前開始ジョブを開始するには、文字ベース・インターフェースを使用します。

コマンド: 事前開始ジョブの開始 (STRPJ)

STRPJ コマンドは、関連するサブシステムの開始が完了するまでは使用しないでください。必要とする事前開始ジョブが正常に開始されるようにするため、STRPJ コマンドが失敗する場合には再試行で遅延ループをコード化します。

同時にアクティブ状態にできる事前開始ジョブの数は、事前開始ジョブ項目の MAXJOBS 属性、およびサブシステムの MAXJOBS 属性によって制限されます。通信項目の MAXACT 属性は、同時に通信項目から保守できるプログラム開始要求の数を制御します。

注: STRJOBS 属性で *NO を指定した場合、サブシステムの開始時に事前開始ジョブ項目に対して事前開始ジョブは開始されません。STRPJ コマンドを実行しても、STRJOBS パラメーターの値は変更されません。

例: 次の例では、サブシステム SBS1 の事前開始ジョブ項目 PJPGM の事前開始ジョブを開始します。サブシステム SBS1 は、このコマンドの発行時にアクティブ状態になければなりません。開始されるジョブ数は、事前開始ジョブ項目 PJPGM の INLJOBS 値で指定された数です。サブシステムは、ライブラリー PJLIB 内のプログラム PJPGM を開始します。

```
STRPJ   SBS(SBS1)  PGM(PJLIB/PJPGM)
```

関連概念

51 ページの『事前開始ジョブ』
事前開始ジョブは、作業要求を受け取る前に実行を開始するバッチ・ジョブです。事前開始ジョブは、サブシステム内の他のタイプのジョブに先立って開始されます。事前開始ジョブは、事前開始ジョブ項目 (サブシステム記述の一部) を使用して、開始時に用いるプログラム、クラス、および記憶域プールを判別するので他のジョブとは異なります。

関連情報

Experience Report: 事前開始ジョブ項目の調整

プログラム開始要求を待ち行列に入れるまたはリジェクトする:

現行の事前開始ジョブ数が事前開始ジョブ項目の MAXJOBS 属性で指定された数よりも少なく、プログラム開始要求の処理に使用できる事前開始ジョブがないときにプログラム開始要求が届くと、この新しい要求をリジェクトするか待ち行列に入れるためのオプションが備えられています。

プログラム開始要求をリジェクトするまたは待ち行列に入れるには、事前開始ジョブ項目の WAIT 属性を使用します。

WAIT(*NO) は、事前開始ジョブがすぐに使用可能でない場合、プログラム開始要求がリジェクトされることを意味します。

WAIT (*YES) は、事前開始ジョブがすぐに使用可能ではなく、MAXJOBS のために、開始してプログラム開始要求を保守できる事前開始ジョブがない場合、プログラム開始要求がリジェクトされることを示します。すぐに使用できる事前開始ジョブがないものの、別の事前開始ジョブを開始できるまたは既に開始されている場合には、プログラム開始要求は待ち行列に入れられます。

次のコマンドは、QGPL ライブラリー内の PGM1 プログラムの事前開始ジョブ項目を、QGPL ライブラリーに含まれる PJSBS サブシステム記述に追加します。QGPL ライブラリー内の PJSBS サブシステムが開始されると、15 個の事前開始ジョブ (QGPL ライブラリー内の PGM1 プログラム) を開始するようこの項目は指定します。事前開始ジョブが使用可能なプールが 4 に減ると (事前開始ジョブが、QGPL ライブラリー内の PGM1 プログラムで指定された要求を保守しているため)、追加の 10 個のジョブが開始されます。要求を受け取る際にこの項目で使用可能な事前開始ジョブがない場合、要求はリジェクトされます。

```
ADDPJE   SBSD(QGPL/PJSBS)  PGM(QGPL/PGM1)  INLJOBS(15)
         THRESHOLD(5)  ADLJOBS(10)  WAIT(*NO)
```

事前開始ジョブ項目の調整:

十分な数の事前開始ジョブをサブシステムで開始しておいて、作業が到着したときに新しいジョブの開始を待機せずに処理されるようにする必要があります。ここでは、最適パフォーマンスのために事前開始ジョブを調整する方法のヒントが示されます。

事前開始ジョブの数の設定:
システムが通常のワークロードを処理し、ワークロードに関する情報を利用できる間に、以下のステップを実行します。

1. サブシステム処理 (WRKSBS) コマンドを使用して、アクティブになっているすべてのサブシステムのリストを取得します。アクティブになっているサブシステムのリストにある各サブシステムごとに、オプション 5 を使用して、サブシステム記述を表示します。

   「サブシステム記述表示」パネルで、オプション 10 を使用して、事前開始ジョブ項目を表示します。そのサブシステム記述に事前開始ジョブ項目がない場合は、WRKSBS リストの次のサブシステムに対して処理を続行します。
2. 「事前開始ジョブ項目の表示 (Display Prestart Job Entries)」パネルで、オプション 5 を使用して、事前開始ジョブ項目の詳細を表示します。「ジョブ初期数」、「しきい値」、および「追加ジョブ数」の現行の設定をメモします。
3. サブシステム記述内の各事前開始ジョブ項目ごとに、活動事前開始ジョブの表示 (DSPACTPJ) コマンドを入力します。 例:

   ```
   DSPACTPJ SBS(SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)
   ```

   DSPACTPJ コマンドが現在許可されていない場合、事前開始ジョブ項目はアクティブではなく、変更する必要はありません。次の事前開始ジョブ項目または次のサブシステム記述に対して処理を続行します。

4. DSPACTPJ 情報を使用して、ワークロードの推定値を取得します。 DSPACTPJ コマンドにより、次のような画面が表示されます。

```
--------------------------------------------------------------------------------
                        活動事前開始ジョブの表示                      SYSTEM
                                                          08/06/03  07:35:00
  サブシステム  . . . . . :   SUBSYSTEM      リセット日付 . . . . . :   08/06/03
  プログラム  . . . . . . :   PJPROGRAM      リセット時刻 . . . . . :   07:23:03
    ライブラリー  . . . . :     PJPGMLIB      経過時間      . . . . :   0000:11:57

事前開始ジョブ:
  現在数      . . . . . . . . . . . . . . . . :   122
  平均数      . . . . . . . . . . . . . . . . :   21.4
  ピーク時の数  . . . . . . . . . . . . . . . :   122

使用中の事前開始ジョブ:
  現在数      . . . . . . . . . . . . . . . . :   120
  平均数      . . . . . . . . . . . . . . . . :   17.7
  ピーク時の数  . . . . . . . . . . . . . . . :   120

                                                                  続く...
続行するには、Enter キーを押してください。

F3= 終了    F5= 最新表示    F12= 取消し    F13= 統計のリセット
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
                        活動事前開始ジョブの表示                      SYSTEM
                                                          08/06/03  07:35:00
  サブシステム  . . . . . :   SUBSYSTEM      リセット日付 . . . . . :   08/06/03
  プログラム  . . . . . . :   PJPROGRAM      リセット時刻 . . . . . :   07:23:03
    ライブラリー  . . . . :     PJPGMLIB      経過時間      . . . . :   0000:11:57
プログラム開始要求:
  待機中の現在数          . . . . . . . . . . . . . :   0
  待機中の平均数          . . . . . . . . . . . . . :   .0
  待機注のピーク時の数    . . . . . . . . . . . . . :   0
  平均待機時間          . . . . . . . . . . . . . . :   00:00:00.0
  受け入れられた数    . . . . . . . . . . . . . . . :   120
  拒否された数      . . . . . . . . . . . . . . . . :   0
                                                                  終わり
続行するには、Enter キーを押してください。

F3= 終了    F5= 最新表示    F12= 取消し    F13= 統計のリセット
--------------------------------------------------------------------------------
```

使用中の事前開始ジョブのセクション、およびピーク数の値を見つけます。この例では、値は 120 です。この数値は、ピーク・ワークロードの推定値です。この値をメモしてください。これは次のステップで使用します。

プログラム開始要求のセクション、およびピーク待機数の値を見つけます。このフィールドを表示するためにページ送りが必要な場合があります。この例では、値は 1 です。この数値は、システムによる新しい作業の到着の処理水準を示します。この値をメモしてください。これは次のステップで使用します。

5. DSPACTPJ を使用したときに、使用中の事前開始ジョブのピーク数がゼロ (0) であることを示す場合、ワークロードで事前開始ジョブ項目が使用されておらず、変更が不要であることを意味します。次の事前開始ジョブ項目または次のサブシステム記述に対して処理を続行します。
6. THRESHOLD パラメーターの値を選択します。使用可能なジョブのプールがこの数より少なくなると、ジョブがさらに開始されます。ジョブの開始には時間がかかります。その間、作業要求がさらに到着する可能性があります。 THRESHOLD には、新規ジョブの開始中に到着する可能性のある要求数に少なくとも 1 を加えた数値を設定してください。

この例で選択された値は 10 です。これは、使用中のジョブのピーク数に基づいて推測した、作業要求の到着の推定値です。これは、入手しにくい測定値の正確な分析ではありません。

前のステップでとったメモをご覧ください。 THRESHOLD の現行の設定が十分高い数値であれば、ピーク待機数はゼロとなります。ピーク待機数がゼロでない場合は、この数を現行の THRESHOLD 値に追加し、その結果を、到着に基づく推定値と比較してください。大きい方の値を使用します。サンプルの DSPACTPJ 情報は値 1 を示していますが、これは THRESHOLD の現行値が低すぎることを意味しています。現行の設定に 1 を加えた数は、推定値の 10 より小さい数です。この例では、値 10 を使用します。

7. ジョブ初期数 (INLJOBS) パラメーターの値を選択します。 INLJOBS は、サブシステムの開始時に開始されるジョブの数を指定します。また、INLJOBS は、あまりにも多くの事前開始ジョブが作業を待機していないかどうかを決定するためにサブシステムが使用するものの一部です。

   前のステップでとったメモをご覧ください。使用中の事前開始ジョブのピーク数をピーク・ワークロードの推定値として使用し、THRESHOLD に値を追加し、その結果を INLJOBS の新しい値として使用します。 DSPACTPJ の情報は、使用中の事前開始ジョブのピーク数が 120 であることを示しています。すでに THRESHOLD として 10 を選択しているので、選択される新しい INLJOBS の値は 130 となります。

8. 追加ジョブ数 (ADLJOBS) パラメーターの値を選択します。 ADLJOBS は、使用可能な事前開始ジョブの数がしきい値 (THRESHOLD) パラメーターで指定した値を下回ったときに開始される事前開始ジョブの追加の数を指定します。

   INLJOBS と THRESHOLD に十分な数値が指定されており、待機要求が発生しないようであれば、ADLJOBS はかなり低くすることができます。 INLJOBS がピーク・ワークロードをかなり下回っている場合は、ADLJOBS を THRESHOLD と同じにする必要があるかもしれません。この例で選択された値は 5 です。

   大きい数値はなるべく使用しないでください。 ADLJOBS に対して大きい値を指定する場合、サブシステムは多数のジョブすべてを 1 度に開始します。これはシステム・パフォーマンスにマイナスの影響を与えることがあり、サブシステムによる他の作業の処理を遅らせてしまいます。

9. 新しく選択した値を、事前開始ジョブ項目で構成した値と比較します。十分な数の事前開始ジョブを確保するには、それぞれのパラメーターの大きい方の値を使用してください。事前開始ジョブ項目変更 (CHGPJE) コマンドを使用して、構成された値を変更します。

   ```
   CHGPJE SBSD(SBSLIB/SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)
          INLJOBS(130) THRESHOLD(10) ADLJOBS(5)
   ```

10. 次の事前開始ジョブ項目または次のサブシステム記述に対して処理を続行します。

詳細

この手順に従う際、いくつかの追加の詳細は良い決定を下す上で役に立つかもしれません。

- THRESHOLD の値が小さすぎると、作業は新しいジョブが開始されるのを待つことになります。場合によっては、要求がタイムアウトになるためにエラーが発生することがあります。

  THRESHOLD が 2 で、作業を待っているジョブが 2 つだけという例があるとします。次の作業要求が到着すると、その要求は待機中のジョブの 1 つに渡され、追加のジョブが開始されます。この例では、新しいジョブの準備が整う前にさらに 2 つの要求が到着します。最初の要求は待機中のジョブによって処理されます。 2 番目の要求は、新しいジョブのいずれかの準備が整うのを待機します。例のワークロ

ードでは、THRESHOLD を少なくとも 3 に設定する必要があります。新しいジョブが開始されている間に到着する要求の数の 2 つに加えて、追加のジョブの作成をトリガーするためにジョブがもう 1 つ必要です。

- サブシステムは必要に応じてジョブを開始するため、必要でないときにジョブを終了することもあります。これは、事前開始ジョブ項目の最大使用数 (MAXUSE) に 1 より大きい数を指定した場合に起きます。 INLJOBS パラメーターの値はサブシステムにジョブの必要数を知らせます。サブシステムがあまりにも多くのジョブを終了しないようにするために、INLJOBS を正しく設定する必要があります。

  INLJOBS の値が小さすぎると、サブシステムはジョブが少なすぎるために定期的に開始し、ジョブが多すぎるために終了します。さらに、システムの使用率が非常に高いときに新しいジョブを開始するためのコストが発生します。
- DSPACTPJ コマンドからのサンプル出力では、使用中の事前開始ジョブのピーク数は 120 で、使用中の事前開始ジョブの平均数は 17.7 となっています。これはピークとしては高い数字ではありません。しかし、平均としては低い数字です。デフォルトで、DSPACTPJ はサブシステムの開始以降発生した事象を示します。平均には、ワークロードがゼロの期間が含まれます。

  F13 を使用して統計をリセットしたり、サンプルのインターバルを注意深く制御したりしたとしても、使用中の事前開始ジョブの平均数は、実際に調整すべき数より小さくなりがちです。ワークロードは平均としておよそ 40 から 60 のジョブを持つことができ、ピークとして 100 から 120 のジョブを持つことができます。

  ピーク・ワークロードの推定値に THRESHOLD を加えた数を INLJOBS に設定するときに、実際のワークロードがピーク・ワークロードの推定値を超過しなければ、サブシステムは追加ジョブを開始する必要がありません。ワークロードのピークが比較的高く、比較的不定期な場合は、INLJOBS に低い方の数を設定することもできます。
- このトピックで示される手順では、標準的な 1 日のピーク・ロードが標準的なピーク・ロードであると想定しています。より多くのデータを収集すると、より正確なワークロードの推定値を生成できるかもしれません。

  ジョブのリスト (QUSLJOB) API またはジョブのリストのオープン (QGYOLJOB) API を使用して、ワークロードを定期的にサンプリングすることができます。いくつかのワークロードは、結果をグラフにすると役に立ちます。事前開始ジョブの数の完全な予測は必要ありません。遅延とタイムアウトを回避するのに十分な予測があれば十分です。
- THRESHOLD および INLJOBS が大きすぎると、不要なアクティブ・ジョブがサブシステムに存在することになります。余分なジョブを開始および終了すると、サブシステムまたは事前開始ジョブ項目を開始および終了するときにさらに時間がかかることになります。

  必要とするより低い値を使用するよりも、若干高い値を使用する方が得策です。ジョブは作業を待機し、メモリーまたはプロセッサーで競合しないので、ジョブを少し多めに持つ分には問題ありません。
- 事前開始ジョブは最初、通信装置で使用されたので、作業の要求はプログラム開始要求と呼ばれ、事前開始ジョブは作業を待機しているとき、PSRW (プログラム開始要求待機中) という状況を示します。

事前開始ジョブのジョブ属性の変更:

大きなジョブ・メッセージ待ち行列はストレージを消費し、ジョブ・ログが大きくなってしまうことがあります (これもまたストレージを消費する)。さらに、IPL の実行中にジョブ・メッセージ待ち行列でリカバリーまたはクリーンアップが必要になると、IPL のパフォーマンスに問題が生じる可能性があります。この例では、事前開始ジョブに関するジョブ・メッセージ待ち行列満杯処置 (JOBMSGQFL) 値およびジョブ・メッセージ待ち行列最大サイズ (JOBMSGQMX) 値を変更する方法を示します。

注: ユーザーに代わってこの一部の処理を行うために、リリース V5R3M0 で QDFTSVR ジョブ記述が導入されました。

他のジョブに影響を与えずに事前開始ジョブのジョブ・メッセージ待ち行列のサイズを制限するには、以下のステップに従います。

1. 対象の事前開始ジョブを見つけ、事前開始ジョブ項目によって使用されるジョブ記述を判別します (これを行うには、サブシステム記述表示 (DSPSBSD) コマンドを使用します)。
2. ジョブ記述が 1 つの事前開始ジョブ項目によってのみ使用されるのか (その場合はそのジョブ記述だけを変更すればよい)、複数の参照、例えばユーザー・プロファイル、事前開始ジョブ項目、他の SBSD 項目などによって使用されるのかを判別します (「分からない」場合のためにジョブ記述を常にもう 1 つ作成しておくこともできますが、既存のジョブ記述に対する変更が対象のジョブだけに影響することが分かっている場合は、その特定のジョブ記述を変更するだけで十分です)。
3. 対象の事前開始ジョブ項目によって使用される新しいジョブ記述を作成します。ジョブ記述作成 (CRTJOBD) コマンドを使用することもできますが、この例では現在使用されているジョブ記述のコピーを作成します。

   注: ジョブ記述 JOBD(*USRPRF) がある場合は、ユーザー・プロファイル表示 (DSPUSRPRF) コマンドを使用して、現在使用されているジョブ記述を判別することができます。デフォルト構成では、ジョブ記述 QDFTJOBD または QDFTSVR が使用されます。

   ```
   DSPUSRPRF USRPRF(QUSER)
   ```

   IBM 提供のオブジェクトとの混同を避けるために、'Q' で始まる名前は使用しません。この例では、事前開始ジョブ項目のジョブ記述の名前として PJJOBD を使用します。オブジェクト複製 (CRTDUPOBJ) コマンドを使用して、QUSER ユーザー・プロファイルによって現在使用されているジョブ記述のコピーを作成します。

   ```
   CRTDUPOBJ OBJ(QDFTSVR) FROMLIB(QGPL) OBJTYPE(*JOBD)
             TOLIB(QGPL) NEWOBJ(PJJOBD)
   ```

4. オブジェクトの所有権と、コピーしたジョブ記述の権限を一致させます。 QDFTSVR と QDFTJOBD は QPGMR が所有しているため、(以下の) 例では、新しく作成されたジョブ記述が QPGMR によって所有されるように変更する方法を示しています。オブジェクト所有者変更 (CHGOBJOWN) コマンドとオブジェクト権限認可 (GRTOBJAUT) コマンドを使用して、オブジェクト所有権と共通権限を正しく設定します。所有者と権限は、オブジェクト権限表示 (DSPOBJAUT) コマンドを使用して見つけることができます。

   ```
   CHGOBJOWN OBJ(QGPL/PJJOBD) OBJTYPE(*JOBD) NEWOWN(QPGMR)

   GRTOBJAUT OBJ(QGPL/PJJOBD) OBJTYPE(*JOBD) USER(*PUBLIC) AUT(*USE)
   ```

5. ジョブ記述変更 (CHGJOBD) コマンドを使用して、ジョブ属性をカスタマイズします。この例では、ジョブ・メッセージ待ち行列の最大サイズとして 8 メガバイトという値を使用します。制限が 64 メガバイトよりかなり小さければ、他の値も使用できます。

   ```
   CHGJOBD JOBD(QGPL/PJJOBD) JOBMSGQMX(8) JOBMSGQFL(*WRAP)
           TEXT('Job attributes for prestart job entries')
   ```

6. システムでアクティブになっているすべての事前開始ジョブ項目に目を通します。サブシステム処理 (WRKSBS) コマンドを使用して、アクティブになっているすべてのサブシステムのリストを取得します。オプション 5 を使用して、サブシステム記述を表示します。オプション 10 を使用して事前開始ジョブ項目を表示し、オプション 5 を使用して事前開始ジョブ項目の詳細を表示します。

   事前開始ジョブ項目が USER(QUSER) および JOBD(*USRPRF) を指定する場合、事前開始ジョブ項目変更 (CHGPJE) コマンドを使用して、新しいジョブ記述を指定します。

```
CHGPJE SBSD(SBSLIB/SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)
           JOBD(QGPL/PJJOBD)
```

事前開始ジョブ項目がジョブ記述を指定する場合、ジョブ記述変更 (CHGJOBD) コマンドを使用して、そのジョブ記述の JOBMSGQMX 値と JOBMSGQFL 値を変更します。

```
CHGJOBD JOBD(JOBDLIB/JOBDNAME) JOBMSGQMX(8) JOBMSGQFL(*WRAP)
```

詳細

QDFTJOBD ジョブ記述はさまざまな事前開始ジョブ項目によって、システムの他のさまざまな場所で使用されます。この例では、PJJOBD という 1 つの新しいジョブ記述を作成しています。新しいジョブ記述はさまざまな事前開始ジョブ項目によって使用されますが、使用する場所は限られています。事前開始ジョブ項目ごとに異なる値を使用するには、各項目ごとに異なるジョブ記述を使用します。事前開始ジョブ項目の中には、事前に固有のジョブ記述が割り当てられているものもあります。

事前開始ジョブのジョブ属性の中には、ジョブ開始時に使用されるジョブ記述から由来していないために、この手順によって変更できないものもあります。事前開始ジョブを使用するサーバーの多くはユーザー・プロファイルをスワップしてから、ジョブ変更 (QWTCHGJB) API を使用してジョブ属性のサブセットを変更します。変更されるジョブ属性は、事前開始ジョブがスワップされたユーザー・プロファイルによって使用されるジョブ記述に由来します。詳しくは、ジョブ変更 API の JOBC0300 フォーマットを参照してください。

いくつかのジョブ属性のジョブ記述は、値がシステム値から取られることを示す場合があります。システム値を変更すると、その変更はジョブ属性をシステム値から取得するすべてのジョブに影響を与えます。ジョブ記述内の値を変更する場合は、ジョブ属性をそのジョブ記述から取得するジョブだけに影響します。

事前開始ジョブの終了:

アクティブ・サブシステムの事前開始ジョブを終了するには、文字ベース・インターフェースを使用できます。

ジョブは要求を待機中であることも、またはすでに要求と関連付けられていることもあります。終了するジョブに関連付けられたスプール出力ファイルも終了できますが、出力待ち行列上に残しておくこともできます。それぞれのジョブ・ログに書き込むメッセージの数に対する制限も変更できます。

注: アクティブ・サブシステムの事前開始ジョブ項目のすべてのジョブを終了するには、事前開始ジョブ終了 (ENDPJ) コマンドを使用します。しかし、問題のある特定の事前開始ジョブのみを終了する場合は、その特定の事前開始ジョブに対してジョブ終了 (ENDJOB) コマンドを使用します。

コマンド: 事前開始ジョブ終了 (ENDPJ)

例: このコマンドは、サブシステム SBS1 内の事前開始ジョブ項目 PJPGM に関連付けられているすべてのジョブを即時に終了させます。これらの事前開始ジョブによって生成されるスプール出力は削除され、ジョブ・ログは保管されます。

```
ENDPJ   SBS(SBS1)  PGM(PJLIB/PJPGM)  OPTION(*IMMED)
        SPLFILE(*YES)
```

例: このコマンドは、サブシステム SBS2 内の事前開始ジョブ項目 PJPGM2 に関連付けられたすべてのジョブを終了させます。これらの事前開始ジョブのスプール出力は、スプール書き出しプログラムによる通常の処理用に保管されます。ジョブは、その即時終了後に、50 秒間ですべてのクリーンアップ・ルーチンを実行します。

```
ENDPJ   SBS(SBS2)  PGM(PJPGM2)  OPTION(*CNTRLD)
        DELAY(50)  SPLFILE(NO)
```

関連概念

51 ページの『事前開始ジョブ』
事前開始ジョブは、作業要求を受け取る前に実行を開始するバッチ・ジョブです。事前開始ジョブは、サブシステム内の他のタイプのジョブに先立って開始されます。事前開始ジョブは、事前開始ジョブ項目 (サブシステム記述の一部) を使用して、開始時に用いるプログラム、クラス、および記憶域プールを判別するので他のジョブとは異なります。

関連情報

Experience Report: 事前開始ジョブ項目の調整

ジョブ・クラス・オブジェクトの管理

クラス・オブジェクトには、ジョブのランタイム環境を制御する実行属性が含まれています。IBM 提供のクラス・オブジェクトまたはクラスは、標準的な対話式アプリケーションおよびバッチ・アプリケーションの両方の必要を満たします。ジョブが使用するクラスは、ジョブを開始するために使用されるサブシステム記述の経路指定項目で指定されます。ジョブが複数の経路指定ステップで構成されている場合、後続のそれぞれの経路指定ステップで使用されるクラスは、経路指定ステップを開始するために使用される経路指定項目の中で指定されます。

クラス・オブジェクトの作成:

クラス・オブジェクトは、文字ベース・インターフェースを使用して作成できます。クラスは、クラスを使用するジョブの処理属性を定義します。ジョブが使用するクラスは、ジョブを開始するために使用されるサブシステム記述の経路指定項目で指定されます。ジョブが複数の経路指定ステップで構成されている場合、後続のそれぞれの経路指定ステップで使用されるクラスは、経路指定ステップを開始するために使用される経路指定項目の中で指定されます。

コマンド: クラス作成 (CRTCLS)

例: この例では、CLASS1 と呼ばれるクラスを作成します。クラスは、ジョブに指定された現行ライブラリーに保管されます。ユーザー・テキスト「This class for all batch jobs from Dept 4836」は、クラスを説明しています。このクラスの属性は、実行優先順位 60、および 900 ミリ秒のタイム・スライスを提供します。ジョブがタイム・スライスの終了時点で実行を終了していない場合、別のタイム・スライスが割り振られるまで、主記憶域外に移される対象になります。残りのパラメーターのデフォルトが想定されます。

```
CRTCLS   CLS(CLASS1)  RUNPTY(60)  TIMESLICE(900)
         TEXT('This class for all batch jobs from Dept 4836')
```

関連概念

35 ページの『クラス・オブジェクト』
クラス・オブジェクトには、ジョブのランタイム環境を制御する実行属性が含まれています。 IBM 提供のクラス・オブジェクトまたはクラスは、標準的な対話式アプリケーションおよびバッチ・アプリケーションの両方の必要を満たします。以下のクラス (名前別) がシステムに提供されます。

クラス・オブジェクトの変更:

クラス・オブジェクトの属性は、文字ベース・インターフェースを使用して変更できます。共通権限権限を除き、どの属性でも変更できます。オブジェクト権限の変更の詳しい情報については、オブジェクト権限取り消し (RVKOBJAUT) コマンドおよびオブジェクト権限付与 (GRTOBJAUT) コマンドを参照してください。

1コマンド: クラス変更 (CHGCLS)

例: このコマンドは、ジョブのライブラリー・リスト上のライブラリーにある、CLASS1 と呼ばれるクラスを変更します。クラスの実行優先順位は 60、および 900 ミリ秒のタイム・スライスに変更されます。

```
CHGCLS   CLS(CLASS1)  RUNPTY(60)  TIMESLICE(900)
```

関連概念

35 ページの『クラス・オブジェクト』
クラス・オブジェクトには、ジョブのランタイム環境を制御する実行属性が含まれています。 IBM 提供のクラス・オブジェクトまたはクラスは、標準的な対話式アプリケーションおよびバッチ・アプリケーションの両方の必要を満たします。以下のクラス (名前別) がシステムに提供されます。

スレッドの管理

スレッドを管理するときに多くのタスクを実行することができます。

特定のジョブの下で実行しているスレッドの表示:

システムで実行中のすべてのアクティブ・ジョブでは、最低 1 つのスレッドが実行されています。スレッドは、ジョブと同じリソースを使用するジョブ内で実行する独立した作業単位です。ジョブはスレッドが実行する作業に依存するので、特定のジョブ内で実行するスレッドを検索する方法を理解することは重要です。

関連概念

38 ページの『スレッド』
スレッド という語は、「制御のスレッド」を省略したものです。スレッドとは、実行中にプログラムがたどるパス、実行されるステップ、およびステップが実行される順序のことです。スレッドは、特定の入力に対して事前定義した順序でコードを開始位置から実行します。

関連情報

例: スレッドを終了する (Java)

Thread management APIs

System i ナビゲーター:
特定のジョブの下で実行しているスレッドを表示するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. 処理したいジョブを右クリックし、「**詳細**」 → 「**スレッド**」をクリックします。

文字ベース・インターフェース:
コマンド: ジョブ処理 (WRKJOB)

例: 以下の例は、ジョブ Crtpfrdta の「スレッドの処理」画面を示しています。

```
WRKJOB JOB(Crtpfrdta) OPTION(*THREAD)
```

スレッドで行えること:

スレッドは、ジョブが実行中に同時に複数の操作を処理するのを支援するため、ジョブ内で実行中のスレッドをモニターすることが必要です。これによって、ジョブが効率的に実行されるようにすることができます。System i ナビゲーター を使用して、管理するスレッドを見付けることができます。

スレッドを見付けたら、そのスレッドを右クリックして以下のいずれかのアクションを選択できます。

統計のリセット
表示しているリスト情報をリセットして、経過時間を 00:00:00 にセットします。

詳細 スレッドの機能はジョブの機能に類似しているので、この 2 つに共通するアクションがいくつかあります。以下のアクションに関する詳細情報があります。

- 呼び出しスタック
- ライブラリー・リスト
- ロック・オブジェクト
- トランザクション
- 経過パフォーマンス統計

保留 スレッドを保留にすることができます。スレッドは、複数回保留にできます。オペレーティング・システムが、スレッドの保留回数を追跡します。

解放 保留にされているスレッドを解放します。スレッドを実行するには、保留になっているスレッドを毎回解放する必要があります。

削除/終了
選択したスレッド (複数可) を終了できます。

スレッド・プロパティー
スレッドのさまざまな属性を表示します。

スレッドで実行できるアクションの詳細については、System i ナビゲーターのオンライン・ヘルプを参照してください。

関連情報

パフォーマンス・システム値: スレッド類縁性

パフォーマンス・システム値: スレッド・リソースの自動調整

スレッド・プロパティーの表示:

スレッドによって、ジョブは一度に複数の処理を行うことができます。スレッドが処理を停止すると、ジョブの実行も停止することがあります。

関連概念

38 ページの『スレッド』
スレッド という語は、「制御のスレッド」を省略したものです。スレッドとは、実行中にプログラムがたどるパス、実行されるステップ、およびステップが実行される順序のことです。スレッドは、特定の入力に対して事前定義した順序でコードを開始位置から実行します。

関連情報

例: スレッドを終了する (Java)

Thread management APIs

System i ナビゲーター:
スレッドの属性を表示するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → **ご使用のシステム** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。
2. 処理したいジョブを右クリックし、「**詳細**」 → 「**スレッド**」をクリックします。
3. 処理したいスレッドを右クリックし、「**プロパティー**」をクリックします。

「一般」タブの情報を使用することにより、スレッドの属性を表示できます。属性としては、スレッド ID、スレッドの詳細な状況、現行ユーザー、実行中のスレッドのタイプ、スレッドの実行対象になっているジョブ、スレッドが実行しているディスク・プール・グループなどがあります。

「パフォーマンス」タブの情報によって、基本的なパフォーマンス情報を表示し、スレッドの実行優先順位を変更することができます。**実行優先順位**は、システムで実行している他のスレッドとの関連におけるスレッドの重要性を示します。可能な値の範囲は、ジョブ優先順位から 99 です (つまり、最高の優先順位は状況によって違います)。スレッドの実行優先順位は、スレッドの実行対象のジョブの実行優先順位よりは高くなりません。

スレッドの開始後に計算されたパフォーマンス (CPU および合計ディスク入出力を含む) を表示できます。また、スレッドに関して計算される**経過パフォーマンス統計**の表示、最新表示、自動最新表示のセットアップ、またはリセットを行うことができます。

文字ベース・インターフェース:
コマンド: ジョブ処理 (WRKJOB)

例: 以下の例は、ジョブ Crtpfrdta の「スレッドの処理」画面を示しています。

```
WRKJOB JOB(Crtpfrdta) OPTION(*THREAD)
```

スレッドの終了または削除:

ジョブの開始時に作成される初期スレッドは、削除または終了することができません。しかし、ジョブが実行を続けられるように 2 次スレッドを終了しなければならないことがあります。終了しようとするスレッドには注意してください。その中で実行しているジョブはそのスレッドの機能がないと完了できないことがあるからです。

重要: スレッドの終了を日次の実行管理ルーチンの一部としないでください。スレッドの終了は他のスレッド内の作業を停止させたり停止させなかったりするので、ジョブの終了よりも重大な事柄です。ジョブを終了すると、すべての作業が停止します。しかし、スレッドを終了すると、作業の一部だけが停止します。他のスレッドは実行を継続する場合もあれば、継続しない場合もあります。終了したスレッドなしで他のスレッドが実行を継続した場合、予期しない結果となることがあります。

2 次スレッドを削除または終了するには、サービス (*SERVICE) 特殊権限またはスレッド制御権限が必要です。

関連概念

38 ページの『スレッド』
スレッド という語は、「制御のスレッド」を省略したものです。スレッドとは、実行中にプログラムがたどるパス、実行されるステップ、およびステップが実行される順序のことです。スレッドは、特定の入力に対して事前定義した順序でコードを開始位置から実行します。

関連情報

例: スレッドを終了する (Java)

Thread management APIs

System i ナビゲーター:

スレッドを削除または終了するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。

2. 作業したいジョブを右マウス・ボタンでクリックして、「**詳細**」をクリックしてから「**スレッド (Threads)**」をクリックします。
3. 終了したいスレッドを右マウス・ボタンでクリックしてから、「**削除/終了**」をクリックします。

文字ベース・インターフェース:
コマンド: ジョブ処理 (WRKJOB)、オプション 20:「**スレッドがアクティブであれば処理**」

例: 以下の例は、ジョブ Crtpfrdta の「スレッドの処理」画面を示しています。

```
WRKJOB JOB(Crtpfrdta) OPTION(*THREAD)
```

「スレッド処理」画面で、オプション: 4=「終了」を選択します。

ジョブ・スケジューリングの管理

Advanced Job Scheduler を使用して実行するジョブをスケジュールするには、System i ナビゲーター の「ジョブのプロパティー」ウィンドウを使用するか、文字ベース・インターフェースを介してジョブ・スケジュール項目を変更します。

System i ナビゲーターを使用したバッチ・ジョブのスケジュール

ジョブのプロパティー -「ジョブ待ち行列」ウィンドウでは、バッチ・ジョブを今すぐ実行する、特定の日時に 1 回実行する、または定期的に (各月の最初の日などに) 実行するようスケジュールするための方法が提供されます。

System i ナビゲーターを使用してジョブをスケジュールするには、以下の指示に従います。

1. 「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**アクティブ・ジョブ待ち行列**」または「**すべてのジョブ待ち行列**」 → **ジョブが入っているジョブ待ち行列**の順に展開します。
2. ジョブを右マウス・ボタンでクリックし、「**プロパティー**」をクリックします。
3. 「ジョブのプロパティー」ウィンドウで、「ジョブ待ち行列」タブをクリックします。
4. ジョブをスケジュールするには、「**ジョブを実行可能にする日時**」の下にあるオプションを使用します。

このウィンドウの使い方については、System i ナビゲーター のヘルプを参照してください。

マネージメント・セントラル・スケジューラーを使用したジョブのスケジュール

プラグイン Advanced Job Scheduler をインストールしていない場合は、マネージメント・セントラル・スケジューラーを使用してジョブをスケジュールできます。

マネージメント・セントラル・スケジューラーを開始するには、System i ナビゲーターの多くのウィンドウに表示される「**スケジュール**」ボタンをクリックします。例えば、System i ナビゲーター の「コマンドの実行」ウィンドウを使用してクリーンアップ・ジョブを投入し、そのジョブがピーク時を過ぎるまで実行しないようにするとします。

1. System i ナビゲーター で、クリーンアップ・ジョブを実行するサーバーを右マウス・ボタン・クリックして、「**コマンドの実行**」をクリックします。
2. 「コマンドの実行」ウィンドウで、ジョブを実行するための文字ベース構文を入力します。援助が必要な場合は、最初のコマンドを入力して「**プロンプト**」をクリックします。
3. コマンドが完了したら、「**スケジュール**」をクリックします。「マネージメント・セントラル・スケジューラー」ウィンドウが表示され、このジョブを 1 回実行するのか、繰り返しジョブとして実行するのかをスケジュールできます。

タスクを 1 回実行するようスケジュールすることもできます。この場合、タスクは指定した日付と時刻に 1 回実行されます。実行されるのが 1 回のみのタスクは、その実行時に「スケジュール・タスク」コンテナーから除去されます。その後、「タスク・アクティビティー」コンテナーに現れます。

重要: マネージメント・セントラル・スケジューラーまたは拡張ジョブ・スケジューラーを使用してジョブをスケジュールした場合、そのジョブを変更または削除するために、「ジョブ・スケジュール項目の操作 (Work with Job Schedule Entries)」(WRKJOBSCDE) を使用しないでください。 WRKJOBSCDE を使用してジョブを変更または削除すると、マネージメント・セントラルには変更が通知されません。 タスクが予想通りに実行されなくなり、マネージメント・セントラルのサーバー・ジョブ・ログにエラー・メッセージが記録される可能性があります。

マネージメント・セントラル・スケジューラーまたは Advanced Job Scheduler を使用してスケジュールされたジョブに対して変更を加える必要がある場合は、System i ナビゲーター・インターフェースを使用します。

関連概念

63 ページの『マネージメント・セントラル・スケジューラー』

System i ナビゲーター には、ジョブを処理するタイミングを編成するための統合されたスケジューラーであるマネージメント・セントラル・スケジューラーが備えられています。 タスクをすぐに実行するか、それとも後で実行するかを選択することができます。マネージメント・セントラル・スケジューラーを使用すると、マネージメント・セントラル内のほとんどのタスクをスケジュールできます。

Advanced Job Scheduler

IBM Advanced Job Scheduler for i5/OS (5761-JS1) ライセンス・プログラムは、1 日 24 時間、週 7 日の不在ジョブ処理が可能な優れたスケジューラーです。このスケジューリング・ツールは、マネージメント・セントラル・スケジューラーよりも多くのカレンダー機能と、スケジュールされたイベントに対するさらに拡張された制御を提供します。ジョブ完了履歴を表示したり、ジョブの状況の通知を管理することもできます。

ネットワーク内の複数のシステムでジョブをする場合は、製品を各システムにインストールする必要があります。 System i ナビゲーター (およびマネージメント・セントラル) の Advanced Job Scheduler を使用する場合、Advanced Job Scheduler がインストールされているシステムからクライアント・プラグインをインストールする必要があります。

ただし、Advanced Job Scheduler ライセンス・プログラムをマネージメント・セントラル・ネットワーク内の各エンドポイント・システムにインストールする必要はありません。 Advanced Job Scheduler をセントラル・システムにインストールすると、エンドポイント・システム上に定義したジョブまたはタスクが、必要なジョブ情報をセントラル・システムから集めるからです。 すべてのジョブ定義情報はセントラル・システムにセットアップしなければなりません。

ネットワーク内の複数のシステムに Advanced Job Scheduler がローカルにインストールされている場合は、マネージメント・セントラル・ネットワークの外でタスクをスケジュールすることになります。 System i ナビゲーターの「**ユーザー接続**」で、「**実行管理機能**」を展開すると、ローカル・システムの Advanced Job Scheduler にアクセスします。

注: 注文情報については、 Job Scheduler for i5/OS  Web サイト (英文) を参照してください。

Advanced Job Scheduler for Wireless:

Advanced Job Scheduler for Wireless は、インターネット電話、PDA Web ブラウザー、または PC Web ブラウザーなど複数のインターネットにアクセス可能な装置上で Advanced Job Scheduler にアクセスできるようにするアプリケーションです。

Advanced Job Scheduler のワイヤレス機能は Advanced Job Scheduler がインストールされているシステム上に存在し、ジョブやアクティビティーにアクセスできるだけでなく、システム上の受信者にメッセージを送信したり、Advanced Job Scheduler モニターを停止および開始することもできます。 Advanced Job Scheduler for Wireless を使用することにより、すべてのユーザーが、独自のブラウズ操作の設定をカスタマイズすることが可能です。たとえば、アクティビティーとジョブを表示するユーザーは、表示されるジョブをカスタマイズできます。

Advanced Job Scheduler for Wireless を使用すると、通常では System i 端末またはエミュレーターにアクセスできない場合でも、ジョブにアクセスすることが可能です。モバイル装置でインターネットに接続し、Advanced Job Scheduler for Wireless サーブレットの URL を入力すればアクセスできます。 この操作を実行すると、メニューが起動し、Advanced Job Scheduler へのリアルタイム・アクセスが実現されます。

Advanced Job Scheduler for Wireless は次の 2 種類の装置で動作します。 1 つは Wireless Markup Language (WML) 装置であるインターネット携帯電話です。もう 1 つは Hypertext Markup Language (HTML) を使用する PDA または PC Web ブラウザーです。このトピックでは、これら 2 つの装置を略して、それぞれ WML と HTML で表します。

Advanced Job Scheduler を使用したジョブのスケジュール:

Advanced Job Scheduler を管理するには、まずライセンス・プログラムをインストールしてから、タスクを実行して Advanced Job Scheduler をカスタマイズしてください。その後、タスクの残りの部分を用いて、このスケジューラーでの作業と管理を行います。

Advanced Job Scheduler のインストール:

マネージメント・セントラル・サーバーへの初回の接続時に、System i ナビゲーター により、Advanced Job Scheduler をインストールするかどうかが尋ねられています。そこでインストールを選択しておらず、インストールを後で実行したい場合は、System i ナビゲーターの「プラグインのインストール (Install Plug-ins)」機能を使用して実行できます。

1. 「**System i ナビゲーター**」ウィンドウで、メニュー・バーから「**ファイル**」をクリックします。
2. 「**インストール・オプション (Install Options)**」 → 「**プラグインのインストール (Install Plug-ins)**」を選択します。
3. Advanced Job Scheduler をインストールする起動システムをクリックして、「**OK**」をクリックします。どの起動システムを使用するか分からない場合は、システム管理者に確認してください。
4. i5/OS の「**ユーザー ID (User ID)**」と「**パスワード (Password)**」に入力して、「**OK**」をクリックします。
5. プラグイン選択リストから「**Advanced Job Scheduler**」をクリックします。
6. 「**次へ (Next)**」をクリックし、さらにもう一度「**次へ (Next)**」をクリックします。
7. 「**完了 (Finish)**」をクリックし、セットアップを完了して終了します。

これで、Advanced Job Scheduler がインストールされました。

スケジューラーの探索:
スケジューラーがある場所を探索するには、以下の手順で行います。

1. 「**マネージメント・セントラル**」を展開します。

2. System i ナビゲーター が新しいコンポーネントを検出したことを示すメッセージが表示されたら、「**スキャンを開始 (Scan Now)**」をクリックします。 「**ユーザー接続**」コンテナーからシステムにアクセスしたときに、このメッセージが表示される場合があります。
3. 「**ユーザー接続**」を展開し、Advanced Job Scheduler ライセンス・プログラムがインストールされているシステムを選択した後、「**実行管理機能**」 → 「**Advanced Job Scheduler**」の順に選択します。

Advanced Job Scheduler のこの予備作業を終えたら、次のステップは Advanced Job Scheduler のセットアップです。

Advanced Job Scheduler のセットアップ:

ジョブのスケジュールを開始する前に、 Advanced Job Scheduler を構成しておく必要があります。

汎用プロパティーの割り当て:

Advanced Job Scheduler で使用する汎用プロパティーを割り当てるには、この指示に従います。 Advanced Job Scheduler のアクティビティー項目とログ項目を保持する期間、およびジョブの実行を許可しない期間を指定できます。

ジョブを処理する作業日、およびスケジュールされた各ジョブにアプリケーションが必要かを指定することもできます。 通知製品がインストールされている場合は、ジョブが完了したときに通知を送信するために使用するコマンドをセットアップしたり、「ジョブ・スケジューラーを使用した配布の送信」(SNDDSTJS) コマンドを使って受信者に通知することができます。

ジョブのアクティビティー記録を保持する期間、およびジョブの実行を許可しない期間を指定できます。ジョブの処理が許可されている作業日、および投入された各ジョブにアプリケーションが必要かを指定することもできます。

通知製品をインストールして、ジョブ終了時に通知 (メッセージ) を受け取れるようにすることも可能です。通知コマンドを定義してジョブが完了または失敗したときに通知を送信することができます。または、「ジョブ・スケジューラーを使用した配布の送信」(SNDDSTJS) コマンドを使って受信者に通知することもできます。

Advanced Job Scheduler 用の汎用プロパティーをセットアップするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右クリックし、「**プロパティー (Properties)**」をクリックします。
3. 「**アクティビティー保持 (Activity Retention)**」を指定します。アクティビティー保持は、ジョブのアクティビティー記録を保持する期間を表します。指定できる値は、1 から 999 までの日数または出現数です。アクティビティーを特定の日数のあいだ保持する場合は「**日数 (Days)**」をクリックし、ジョブごとに特定の出現数のあいだアクティビティーを保持する場合は「**ジョブごとの出現数 (Occurrences per job)**」をクリックします。
4. 「**ログ保持 (Log retention)**」を指定する。ログ保持は、Advanced Job Scheduler のログ項目を保持する期間を日数で指定します。
5. 「**予約済み期間 (Reserved period)**」を指定することもできます。この期間中は、ジョブは実行されません。
6. リストから作業日を指定する。選択した日は作業日として指定され、 ジョブをスケジュールするときに参照されます。
7. 「**スケジュールしたジョブに必要なアプリケーション (Application required for scheduled job)**」をクリックして、スケジュールしたジョブごとにアプリケーションが必要かどうかを指定する。**アプリケー**

ションとは、処理のために複数のジョブが 1 つにまとめられたものを指します。既存のジョブにアプリケーションが含まれていない場合、このフィールドは選択できません。特定のジョブのためにアプリケーションを使用することにした場合は、アプリケーションを使用した作業に進んでください。

8. 「**カレンダー (Calendars)**」をクリックして、使用するスケジューリング・カレンダー、祝祭日カレンダー、会計カレンダーをセットアップする。
9. 周期的に実行するようスケジュールされているジョブについて、開始時刻を起点にして次の実行時間を設定する場合は、「**実行周期の起点を開始時刻にする (Base periodic frequency on start time)**」をクリックします。例えば、あるジョブが午前 8:00 から 30 分ごとに実行され (24 時間通してジョブを実行する場合は、終了時刻に午前 7:59 と指定します)、ジョブの実行時間は合計で 20 分であるとします。このフィールドをチェックすると、ジョブは午前 8:00、午前 8:30、午前 9:00、以降 30 分ごとに実行されます。 このフィールドをチェックしない場合は、ジョブは午前 8:00、午前 8:50、午前 9:40、午前 10:30、以降 50 分ごとに実行されます。
10. 「**保留ジョブのリセット (Reset held jobs)**」をクリックして、再計算を続行し、保留ジョブを実行する次の日時を表示します。
11. 「**1 日の開始時刻 (Start time of day)**」を指定する。この時刻が、1 日の始まりと見なされます。この時刻を使用するように指定されたすべてのジョブについて、ジョブの開始する時刻が「**1 日の開始時刻 (Start time of day)**」フィールドの時刻より前であると、そのジョブ日付は一日前に変更されます。
12. 「**ジョブ・モニター・ユーザー (Job monitor user)**」を指定する。このフィールドは、モニター・ジョブの所有者として使用するユーザー・プロファイルの名前を指定します。「**現行ユーザー (Current user)**」が指定されているすべてのジョブは、モニター・ジョブのユーザー・プロファイルを使用します。モニター・ジョブのデフォルト・ユーザー・プロファイルは QIJS です。
13. 「**通知コマンド (Notification command)**」フィールドに、コマンドを指定する。システムに提供されている「ジョブ・スケジューラー (通知) を使用した配布の送信 (SNDDSTJS)」コマンドを使用するか、通知ソフトウェアによって指定されているコマンドを使用します。 SNDDSTJS コマンドは、Advanced Job Scheduler の通知機能を使用します。 指定された受信者は、ジョブ・スケジュール項目の完了が正常または異常であることを示すメッセージを受け取ることができます。

許可レベルの指定:

この情報では、ジョブ (つまり製品の諸機能) に許可レベルを指定し、新規ジョブのデフォルト許可を設定する方法を示します。

ジョブ (つまり製品の機能) に許可レベルを指定し、新規ジョブのデフォルト許可を設定して各「ジョブ制御/アプリケーション (Job Control/Application)」に関連付けることができます。ジョブの許可を使用すると、投入、管理、許可、表示、コピー、更新、または削除といったアクションへのアクセスを付与または拒否できます。「スケジュール・カレンダーの処理」、「報告書の送信」、および「ジョブの追加」など製品の個々の機能へのアクセスも付与または拒否できます。

新規ジョブを追加すると、そのジョブにデフォルト許可レベルが転送されます。この場合、システムはジョブ定義に指定されたアプリケーションに基づいて「新規ジョブ (New Job)」許可を転送します。 アプリケーションを使用しない場合、システムは「*SYSTEM 新規ジョブ (*SYSTEM New Job)」許可を転送します。

製品の各機能の許可レベルの指定:

製品の各機能に許可レベルを指定するには、以下の手順で行います。

1. System i ナビゲーターで、「**実行管理機能**」を展開します。

2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**許可 (Permissions)**」をクリックします。
4. 機能を選択して、「**プロパティー (Properties)**」をクリックします。
5. 「機能の許可プロパティー (Function Permissions Properties)」ウィンドウで、必要に応じて許可レベルを編集します。共通または特定ユーザーへのアクセスを付与または拒否することができます。

ジョブへの許可レベルの指定:
ジョブに許可レベルを指定するには、以下の手順で行います。
1. System i ナビゲーターで、「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**スケジュール済みジョブ (Scheduled Jobs)**」をクリックします。
3. スケジュール済みジョブを右クリックして、「**許可 (Permissions)**」をクリックします。
4. 「許可プロパティー (Permissions Properties)」ウィンドウで、 必要に応じて許可レベルを編集します。共通または特定ユーザーへのアクセスを付与または拒否することができます。また、投入、管理、許可、表示、コピー、更新、または削除の許可を指定できます。

デフォルト許可レベルの指定:
ジョブ制御/アプリケーションに関連付けられた新規ジョブにデフォルト許可レベルを指定するには、以下の手順で行います。
1. System i ナビゲーターで、「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**ジョブ制御/アプリケーション (Job Controls/Applications)**」をクリックします。
4. リストからジョブ制御またはアプリケーションを選択して、「**新規ジョブ許可 (New Job Permissions)**」をクリックします。
5. 「機能の許可プロパティー (Function Permissions Properties)」ウィンドウで、必要に応じて許可レベルを編集します。共通または特定ユーザーへのアクセスを付与または拒否することができます。また、投入、管理、許可、表示、コピー、更新、または削除の許可を指定できます。

スケジューリング・カレンダーのセットアップ:

この説明では、ジョブまたはジョブ・グループをスケジュールするために、選択した日のカレンダーをセットアップする方法を示します。このカレンダーは、日付を指定してジョブをスケジュールしたり、他のスケジュールと連動させて使用することができます。

スケジューリング・カレンダーとは、ジョブまたはジョブ・グループをスケジュールするために使用する、選択した日のカレンダーのことです。複数のスケジューリング・カレンダーの表示、新規スケジューリング・カレンダーの追加、既存のスケジューリング・カレンダーに基づいた新規スケジューリング・カレンダーの追加、および現在スケジュールされているジョブで使用されていない既存のカレンダーの除去を行うことができます。

カレンダーを選択し、そのプロパティーを表示して変更することができます。カレンダーを選択すると、そのカレンダーの詳細情報が「詳細 (Details)」に表示されます。

スケジューリング・カレンダーをセットアップするには、以下の手順で行います。
1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。

2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「汎用 (General)」ページで、「**カレンダー (Calendars)**」をクリックします。
4. 「スケジューリング・カレンダー (Scheduling Calendars)」ページで、「**新規 (New)**」をクリックします。
5. 「**名前 (Name)**」を指定します。
6. 「**説明 (Description)**」フィールドに、カレンダーについて説明するテキストを指定します。
7. 「**参照カレンダー (Reference calendar)**」を選択します (該当する場合)。このカレンダーはすでにセットアップ済みのカレンダーで、そのプロパティーが新規カレンダーにマージされているかのように適用されます。 Advanced Job Scheduler を初めて使用する場合、参照カレンダーはありません。
8. カレンダーに組み込む日付を選択します。選択した日付ごとにそれが今年の日付かそれとも毎年の日付かを「**選択した日付 (Selected date)**」フィールドに指定してから、別の日付をカレンダーに追加することができます。その指定を行わないと、別の日付をクリックしたときに、前に選択した日付が選択を解除されてしまいます。
9. カレンダーに特定の曜日を組み込む場合は、そのように指定します。

祝祭日カレンダーのセットアップ:

この説明では、スケジュールされたジョブの処理を許可しない日のカレンダーをセットアップする方法を示します。例外日ごとに代替日を指定したり、例外日には処理を完全にスキップさせることもできます。

祝祭日カレンダーとは、Advanced Job Scheduler のジョブを処理しない日を指定する例外カレンダーのことです。祝祭日カレンダーに指定する例外日には、それぞれに代替日を指定することができます。複数の祝祭日カレンダーの表示、新規祝祭日カレンダーの追加、既存の祝祭日カレンダーに基づいた新規祝祭日カレンダーの追加、および現在スケジュールされているジョブで使用されていない既存のカレンダーの除去を行うことができます。

祝祭日カレンダーでは、事前定義スケジュールを使用できます。たとえば、毎月第 3 金曜日という頻度を表すスケジュール THIRDFRI を作成したとします。祝祭日カレンダーに THIRDFRI を使用すると、この祝祭日カレンダーを使用するすべてのジョブは毎月第 3 金曜日には実行されなくなります。 1 つの祝祭日カレンダーで、1 つ以上のスケジュールを使用することが可能です。スケジュールによって生成された日付は、カレンダー上で黒い枠で囲まれて表示されます。

カレンダーを選択し、そのプロパティーを表示して変更することができます。カレンダーを選択すると、そのカレンダーの詳細情報が「詳細 (Details)」に表示されます。

祝祭日カレンダーのセットアップ:
祝祭日カレンダーをセットアップするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」を選択します。
3. 「汎用 (General)」ページで、「**カレンダー (Calendars)**」をクリックします。
4. 「**祝祭日カレンダー (Holiday Calendars)**」タブをクリックします。
5. 「**新規 (New)**」をクリックして、カレンダーの名前を入力します。
6. 「**説明 (Description)**」フィールドに、カレンダーについて説明するテキストを指定します。
7. 「**参照カレンダー (Reference calendar)**」を選択します (該当する場合)。このカレンダーはすでにセットアップ済みのカレンダーで、そのプロパティーが新規カレンダーにマージされているかのように適用されます。 Advanced Job Scheduler を初めて使用する場合、参照カレンダーはありません。

8. カレンダーに組み込む日付を選択します。選択した日付ごとにそれが今年の日付かそれとも毎年の日付かを「**選択した日付 (Selected date)**」フィールドに指定してから、別の日付をカレンダーに追加することができます。その指定を行わないと、別の日付をクリックしたときに、前に選択した日付が選択を解除されてしまいます。
9. ジョブを実行する代替日を選択します。直前の作業日、翌作業日、または特定の日付を選択することができますが、何も選択しなくてもかまいません。特定の日付を選択するには、「**特定の代替日 (Specific alternate date)**」をクリックして、 日付を入力します。
10. カレンダーに組み込む特定の曜日を指定します。

祝祭日カレンダーへのスケジュールの追加:

スケジュールされたジョブに祝祭日カレンダーを追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**汎用 (General)**」ページで、「**カレンダー (Calendars)**」をクリックします。
4. 「祝祭日カレンダー (Holiday calendar)」ページで、祝祭日カレンダーを選択して、「**プロパティー (Properties)**」をクリックします。
5. このタブ・ページの左下にある「**スケジュール (Schedules)**」をクリックします。
6. 適切なスケジュールを選択して、「**追加 (Add)**」をクリックします。
7. **代替日**を変更するには、「**選択済みスケジュール (Selected Schedules)**」リストからスケジュールを右マウス・ボタン・クリックし、正しい**代替日**をクリックします。

会計カレンダーのセットアップ:

会計年度を月以外の期間に分ける場合、このステップに従って、ジョブまたはジョブ・グループをスケジュールするために、選択した日の会計カレンダーをセットアップします。

*会計カレンダー*とは、ジョブまたはジョブ・グループをスケジュールするために使用する、選択した日のカレンダーのことです。業務に固有の会計カレンダーを定義する場合に、会計カレンダーを使用します。会計年度の各期の開始日と終了日を指定できます。

会計カレンダーをセットアップするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「汎用 (General)」ウィンドウで、「**カレンダー (Calendars)**」をクリックします。
4. 「会計カレンダー (Fiscal Calendars)」ページで、「**新規 (New)**」をクリックします。
5. 「**名前 (Name)**」を指定します。
6. 「**説明 (Description)**」フィールドに、カレンダーについて説明するテキストを入力します。
7. 「**会計カレンダーのプロパティー (Fiscal Calendar Properties)**」ウィンドウで「新規 (New)」をクリックして、新規項目を作成します。
8. 期間を選択し、開始日と終了日を指定します。期間は 13 個まで指定できます。
9. 「**OK**」をクリックして、会計カレンダーの項目を保存します。
10. 7 から 9 までのステップを必要に応じて繰り返す。

通知のために使用するメール・サーバーの指定:

E メール通知メッセージを送信するには、メール・サーバーが必要です。

通知プロパティーをセットアップするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**通知 (Notification)**」を右マウス・ボタン・クリックし、「**プロパティー (Properties)**」をクリックします。
4. メッセージを保管する日数を指定します。日数は「**メッセージ保持 (Message retention)**」フィールドに指定します。
5. 「**発信メール・サーバー (SMTP) (Outgoing mail server (SMTP))**」を指定します。たとえば、SMTP.yourserver.com のように指定します。
6. 「**ポート (Port)**」を指定します。デフォルトのポート番号は 25 です。
7. E メール・アドレスを「**返信アドレス (Reply address)**」フィールドに指定します。すべての返信メッセージはこのアドレスに送信されます。
8. 「**送信アクティビティーをログに記録する (Log send activity)**」フィールドで「**はい (Yes)**」または「**いいえ (No)**」を選択します。送信アクティビティーは、問題判別に使用されます。
9. 許可されている「**バナー・ページの数 (Number of banner pages)**」を指定します。これは「報告書配布 (Report Distribution)」で使用されます。
10. 「**OK**」をクリックして、通知プロパティーを保管します。

複数スケジューリング環境のセットアップ:

同一のシステム上に複数のスケジューリング環境をセットアップすることができます。これにより、元のデータ・ライブラリーをアクティブ・データ・ライブラリーとして動作させ、コピーしたデータ・ライブラリーをテスト用に使用することができます。したがって、テスト用と実動用の 2 つのスケジューリング環境を持つことになります。また、元のシステムにシステム障害が起きたときには、テスト・データ・ライブラリーがバックアップの役割をも果たします。この機能により、元のデータ・ライブラリーでエラーが起きたとしても、データ・ライブラリーのバックアップ・コピーがあるおかげで、保護が強化されます。

複数スケジューリング環境をセットアップする理由には、いくつかが挙げられます。製品の実動バージョンとテスト・バージョンを同時に実行できる環境を希望するかもしれません。このような環境があれば、さまざまなジョブ・スケジュールをテストしてから、実動システムのデータ・ライブラリーでジョブ・スケジュールを実際に使用することができます。あるいは、あるシステムを 1 つ以上の他のシステムのバックアップにし、そのシステムでデータ・ミラーリング製品を使用して、Advanced Job Scheduler データ・ライブラリー (QUSRIJS) を ソース・システムから別の名前のライブラリーに複製することができます。この場合は、ソース・システムに問題が起きない限りデータ・ライブラリーはアクティブです。

スケジューリング環境は QUSRIJS ライブラリーの複製で、データだけが異なります。たとえば、QUSRIJSTST という名前の別のデータ・ライブラリーに QUSRIJS のようなオブジェクトを入れることができます。それぞれがデータ・ライブラリーとみなされます。

複数スケジューリング環境をセットアップするには、以下の手順で行います。

1. システムからデータ・ライブラリーを入手する

   データ・ライブラリーを作成するには、システムからデータ・ライブラリーを入手することが必要です。システムからデータ・ライブラリーを入手する方法には、以下の 3 通りがあります。

   - システムからデータ・ライブラリーを保管して、それを実動システムに復元する。

- 「ライブラリーのコピー (CPYLIB)」コマンドを使用して、現行システム上にデータ・ライブラリーを複製する。
- テスト・システム上にデータ・ライブラリーをミラーリングする。実動システムとテスト・システムで実行されているバージョンとリリースのレベルが同じであるようにします。

注: コピー、復元、またはミラーリングされたデータ・ライブラリーの名前は、元のシステムで使用されているものとは異なる名前にします。

2. ユーザーへのデータ・ライブラリーの割り当て

   テスト・データ・ライブラリーを取得したら、そのデータ・ライブラリーを Advanced Job Scheduler のプロパティーに追加して、データ・ライブラリーにユーザーを割り当てます。それによって、ユーザーが Advanced Job Scheduler を使用する際には、ユーザーが加えた変更はそのユーザーに割り当てられたデータ・ライブラリーに保管されます。

3. テスト・データ・ライブラリーから実際のデータ・ライブラリーへのジョブのコピー (オプション)

   テストのためにデータ・ライブラリーを使用している場合に、ジョブをテスト・データ・ライブラリーから実際に使用されているデータ・ライブラリーにコピーしたいと思うことがあります。ステップ 1 でデータ・ライブラリーを復元またはコピーしており、かつ実際に使用されているデータ・ライブラリーに移動したいジョブがある場合には、ジョブをコピーするだけで十分です。データ・ライブラリーが実動システムからテスト・システムにミラーリングされている場合は、コピーする必要がありません。

   あるシステムのデータ・ライブラリーから別のシステムのデータ・ライブラリーにジョブをコピーするには、「ジョブ・スケジューラーを使用したジョブのコピー」(CPYJOBJS)コマンドを使用します。このコマンドで使用される特定のパラメーターについて詳しくは、 オンライン・ヘルプを参照してください。

ユーザーへのデータ・ライブラリーの割り当て:

割り当てられたデータ・ライブラリーには、ユーザーが Advanced Job Scheduler を使用して加えた変更がすべて保管されます。 データ・ライブラリーには、QUSRIJS ライブラリーで検出されたすべてのオブジェクトが含まれます。持つことができるデータ・ライブラリーの数に制限はありません。

データ・ライブラリーをユーザーに割り当てるには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右クリックし、「**プロパティー (Properties)**」をクリックします。
3. 「データ・ライブラリー (Data Libraries)」ウィンドウで、「**追加 (Add)**」をクリックしてデータ・ライブラリーを指定します。リストされたデータ・ライブラリーは、システム上のすべてのユーザーが使用可能です。
4. 「ユーザー (Users)」ウィンドウで、「**追加 (Add)**」をクリックして新規ユーザーを追加します。
5. 名前を指定します。
6. データ・ライブラリーを選択します。
7. 「**OK**」をクリックして、ユーザーを追加します。
8. 「**プロパティー (Properties)**」をクリックして、 ユーザーに割り当てられているデータ・ライブラリーを変更します。

データ・ライブラリーを使用すると、複数スケジューリング環境のセットアップを行うことができます。

Advanced Job Scheduler の管理:

この情報では、Advanced Job Scheduler を使用してジョブをスケジュールする方法を示します。

ジョブの作成およびスケジュール:

ジョブをスケジュールし、そのジョブに関連付けるコマンドを指定することができます。また、開始コマンドと終了コマンドを指定して、特別にスケジュールしたジョブを実行することもできます。

新規スケジュール済みジョブを作成およびスケジュールするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックします。
3. 「**スケジュール済みジョブ (Scheduled Job)**」を右マウス・ボタン・クリックして、「**新規スケジュール済みジョブ (New Scheduled Job)**」をクリックします。

ジョブ・グループの作成およびスケジュール:

一連のジョブを指定した順序で連続して実行するようにセットアップおよびスケジュールすることができます。同じジョブ・グループ内では、必ず 1 つのジョブが完了してから、次に投入されたジョブが処理されるようにします。

ジョブ・グループとは、指定された順序で連続して実行するためにグループにまとめられたジョブのことです。グループ内の各ジョブが正常に完了してはじめて、次のジョブの処理が投入されます。グループ内のジョブが 1 つでも正常に完了しないと、そのグループの処理は停止してしまいます。

新規ジョブ・グループを作成およびスケジュールするには、以下のようにします。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」をクリックします。
3. 「**ジョブ・グループ (Job Groups)**」を右マウス・ボタン・クリックし、「**新規ジョブ・グループ (New Job Group)**」をクリックします。

新規ジョブ・グループの詳細を入力するにあたって、詳しい情報はオンライン・ヘルプを参照してください。

事前定義スケジュール:

ジョブをスケジュールしたり、祝祭日カレンダー内の例外日を計算したりするために必要な情報を含むスケジュールを作成することができます。

たとえば、追加のカレンダーに加えて、実行する曜日を含む ENDOFWEEK スケジュールを作成することができます。作成された ENDOFWEEK スケジュールは、スケジューリング頻度が一致するすべてのジョブによって使用されます。この機能には System i ナビゲーター からしかアクセスできません。

ジョブで使用されているのと同じ事前定義スケジュールを、祝祭日カレンダーで使用することができます。たとえば、毎月第 3 金曜日という頻度を表すスケジュール THIRDFRI を作成したとします。祝祭日カレンダーに THIRDFRI を使用すると、この祝祭日カレンダーを使用するすべてのジョブは毎月第 3 金曜日には実行されなくなります。 1 つの祝祭日カレンダーで、1 つ以上のスケジュールを使用することが可能です。スケジュールによって生成された日付は、カレンダー上で黒い枠で囲まれて表示されます。

事前定義スケジュールのセットアップ:

事前定義スケジュールをセットアップするには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。

2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**スケジュール (Schedules)**」タブをクリックします。
4. 「**新規 (New)**」をクリックして、スケジュールの名前を入力します。
5. スケジュールの説明を入力します。
6. スケジュールに含める頻度と日付を、追加のカレンダーとともに (もしあれば) 選択します。

新規スケジュールの詳細を入力するにあたって、詳しい情報はオンライン・ヘルプを参照してください。

スケジュール済みジョブへのスケジュールの追加:

スケジュール済みジョブにスケジュールを追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックします。
3. 「**スケジュール済みジョブ (Scheduled Jobs)**」をクリックして、ジョブをリストします。
4. スケジュール済みジョブを右マウス・ボタンでクリックして、「**プロパティー (Properties)**」をクリックします。
5. 「**スケジュール (Schedule)**」タブをクリックします。
6. タブの右上から、適切な「スケジュール (Schedule)」オプションを選択します。

祝祭日カレンダーへのスケジュールの追加:

祝祭日カレンダーとは、Advanced Job Scheduler のジョブを処理しない日を指定する例外カレンダーのことです。祝祭日カレンダーに指定する例外日には、それぞれに代替日を指定することができます。

祝祭日カレンダーにスケジュールを追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**汎用 (General)**」ページで、「**祝祭日カレンダー (Holiday Calendars)**」をクリックします。
4. 「祝祭日カレンダー (Holiday Calendars)」ページで、祝祭日カレンダーを選択して、「**プロパティー (Properties)**」をクリックします。
5. このタブ・ページの左下にある「**スケジュール (Schedules)**」をクリックします。
6. 適切なスケジュールを選択して、「**追加 (Add)**」をクリックします。
7. **代替日**を変更するには、「**選択済みスケジュール (Selected Schedules)**」リストからスケジュールを右マウス・ボタン・クリックし、正しい**代替日**をクリックします。

詳しくは、オンライン・ヘルプを参照してください。

一時スケジュール・ジョブの作成:

通常のスケジュールに加えて、スケジュールされたジョブを現時点でまたは将来に実行することが必要になるときがあります。 「ジョブの処理」画面のオプション 7 「ジョブ・スケジューラーを使用したジョブの投入 (SBMJOBJS)」コマンド、または System i ナビゲーター から「**実行 (Run)**」オプションを使用します。この場合の特殊な実行をセットアップするにあたって、コマンド・リストの一部のコマンドだけを処理するようにすることが必要になる場合があります。

SBMJOBJS コマンドでは、開始コマンド・シーケンスと終了コマンド・シーケンスを指定することができます。 たとえば、JOBA には 5 つのコマンドがあり、シーケンスは 10 から 50 までであるとします。SBMJOBJS コマンドに、シーケンス 20 で開始してシーケンス 40 で終了するよう指定します。 このとき、シーケンス 10 と 50 はバイパスされます。

System i ナビゲーター では、コマンド・リスト内で開始コマンドと終了コマンドを選択できます。

System i ナビゲーター を使用して特別にスケジュールしたジョブを実行するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックします。
3. 「**スケジュール済みジョブ (Scheduled Jobs)**」をクリックして、ジョブをリストします。
4. スケジュール済みジョブを右マウス・ボタンでクリックし、「**実行(Run)**」をクリックします。
5. ジョブを現時点または将来実行するかどうかを指定します。
6. 開始コマンドと終了コマンドを選択します。

新規ジョブの詳細を入力するにあたって、 詳しい情報はオンライン・ヘルプを参照してください。

ジョブ依存関係のスケジュール:

Advanced Job Scheduler を使用することにより、現在の環境でジョブが処理される方法を反映した依存関係をセットアップすることができます。 依存関係によって、ジョブまたはジョブ・グループがいつ実行されるかを判別できます。ジョブを実行する前にすべての依存関係を集めるようにするか、あるいはジョブを実行する前に少なくとも 1 つの依存関係があるようにすることが可能です。

依存関係には、以下の種類があります。

- **ジョブ依存関係**

  ジョブ依存関係とは、ジョブどうしの間で先行ジョブまたは後続ジョブの関係を表すものです。先行ジョブとは、後続のジョブが実行する前に実行されるジョブのことです。 後続ジョブとは、すべての先行ジョブの処理が済んでから実行されるジョブのことです。 1 つの先行ジョブに対して複数の後続ジョブがある場合があれば、1 つの後続ジョブに対して複数の先行ジョブがある場合もあります。また、ある従属ジョブが実行しないスケジュールになっている日に、そのジョブの先行ジョブと後続ジョブが実行する場合には、 従属ジョブをスキップするように指定することができます。

- **アクティブ依存関係**

  アクティブ依存関係とは、選択したジョブが投入されるまでアクティブにすることができないジョブのリストのことです。リストにあるジョブのいずれかがアクティブであると、Advanced Job Scheduler は選択したジョブが実行されないようにします。選択したジョブは、リスト内のすべてのジョブが非アクティブになるまで遅延されます。

- **リソース依存関係**

  リソース依存関係は、いくつかの点に基づいています。以下に各タイプを示し、チェックされる分野について説明します。リソース依存関係のタイプは、以下のとおりです。

  ファイル

  ジョブは、ファイルが存在しているかいないか、またそのファイルを処理するために指定された割り振りレベルと一致しているかどうかの影響を受けます。また、ジョブを処理する前に、レコードがあるかどうかもチェックされます。たとえば、ファイル ABC が存在しており、かつその

ファイルを排他的に割り振ることができ、かつそのファイル内にレコードがあるという場合にのみ、JOBA が実行されるように JOBA をセットアップすることができます。

オブジェクト
ジョブは、QSYS タイプのオブジェクトが存在しているかいないか、またそのオブジェクトを処理するために指定された割り振りレベルと一致しているかどうかの影響を受けます。たとえば、データ域 XYZ がある場合にのみ、JOBA が実行されるように JOBA をセットアップすることができます。ジョブは、統合ファイル・システム内で検出されるオブジェクトが存在しているかいないかの影響を受ける場合があります。依存関係がパス内のいずれかのオブジェクトに基づいている場合、統合ファイル・システムのパスの末尾をスラッシュ '/' にします。

ハードウェア構成
ジョブは、ハードウェア構成が存在しているかいないか、および処理するハードウェア構成の状況の影響を受けます。たとえば、装置 TAP01 があり、構成状況が「使用可能 (Available)」である場合にのみ、JOBA が実行されるように JOBA をセットアップすることができます。

ネットワーク・ファイル
ジョブが処理されるかどうかは、ネットワーク・ファイルの状況の影響を受けます。

サブシステム
ジョブが処理されるかどうかは、サブシステムの状況の影響を受けます。

ジョブ依存関係を処理するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**スケジュール済みジョブ (Scheduled Jobs)**」をクリックします。
4. 依存関係を処理する「**ジョブ名 (Job Name)**」を右マウス・ボタン・クリックします。
5. 「**ジョブ依存関係 (Job Dependencies)**」、「**アクティブ依存関係 (Active Dependencies)**」または「**リソース依存関係 (Resource Dependencies)**」のいずれかを選択します。詳しくは、オンライン・ヘルプを参照してください。

Work Flow Manager:

Work Flow Manager によって、自動ステップまたは手動ステップで構成される作業単位を定義できるようになりました。これらの作業単位は、スケジュールすることも対話式に実行することもできます。 Work Flow Manager は、System i ナビゲーター インターフェースの Advanced Job Scheduler コンテナー内にあります。

ワークフロー内の各ステップは、1 つ以上の先行 Advanced Job Scheduler ジョブ、および 1 つ以上の後続 Advanced Job Scheduler ジョブを持つことができます。ワークフローの開始時には、最初のステップに実行のフラグが立てられます。それが完了すると、次のステップに実行のフラグが立てられ、これが繰り返されます。

以下に示すのは、Work Flow Manager を使用する場合の付加的ないくつかの考慮事項です。

- どのステップでもワークフローを手動で開始できます。そのようにする場合、ワークフロー内の前のすべてのステップはバイパスすることになります。
- 自動ステップは、前のすべてのステップが完了した後で完了します。これには、すべての先行 Advanced Job Scheduler ジョブが含まれます。
- ステップが完了した後で、後続 Advanced Job Scheduler ジョブに実行のフラグが立てられます。
- 手動ステップは、ステップの先行ジョブが終了していれば、任意の順序で完了できます。

- 後続の完了していない自動ステップがなければ、完了している手動ステップを、未完了とマークして再度実行することができます。
- 前のステップの後続ジョブと同じ先行ジョブを指定することにより、ステップにジョブが完了するまで待機させて、それからステップの完了を通知させることができます。
- 特定のステップについて、その開始時、停止時、特定の時点で開始しなかった、時間がかかりすぎているなどの場合に他のユーザーに通知することができます。たとえば、特定の手動ステップを担当しているユーザーに、前の自動化ステップが完了していることを通知できます。

ワークフローを使用する場合、アクティビティー・ログは、ワークフローの開始時、実行されたステップ、自動化ステップの状況 (成功または失敗)、ワークフローの終了時、およびワークフローの最終状況を表示します。

表 8. ワークフローの例

ワークフロー	**給与計算**
スケジュール済み	毎週金曜日の午後 1:00
通知	クラーク - 給与計算ワークフローは開始しています。
ステップ 1	自動 - 給与計算ファイルを初期化する後続ジョブを指定します。
ステップ 2	自動: • ステップ 1 からの後続ジョブをこのステップの先行ジョブとして指定します。 • クラークにタイムカードを入力できることを通知します。
ステップ 3	手動: • クラークはタイムカードの入力後に完了します。 • タイムカード・ファイルを処理し、タイムカード・レポートを印刷する後続ジョブを指定します。 • ステップが 120 分以内に完了しない場合はスーパーバイザーに通知します。
ステップ 4	自動: • 前のステップからの後続ジョブを先行ジョブとして指定します。 • 後続ジョブなし • クラークにタイムカード報告書を検査するように通知します。
ステップ 5	手動: • クラークは報告書の検査後に完了します。 • 後続ジョブが給与計算を処理するように指定します。
ステップ 6	自動: • 前のステップからの後続ジョブを先行ジョブとして指定します。 • 後続ジョブなし • クラークおよびスーパーバイザーに給与計算が完了したことを通知します。

この例では、ワークフロー PAYROLL は各金曜日の 1:00 p.m. に開始します。ワークフローが開始したことがクラークに通知されます。

ステップ 1 は自動であり、先行ジョブはないので、給与計算ファイルを初期化する後続ジョブに実行のフラグを立て、次いで完了させます。ステップ 2 は、ステップ 1 の後続ジョブを、その先行ジョブとして持ちます。ステップ 2 は、給与計算ファイルを初期化するジョブの完了を待機します。その完了後に、ステップ 2 はクラークに、タイムカードを入力できることを通知します。実行のフラグが立てられている後続ジョブはありません。

すべてのタイムカードの入力後に、クラークは手動でステップ 3 を完了します。タイムカード・ファイルを処理し、タイムカード報告書を印刷する後続ジョブに、実行のフラグが立てられます。予防措置として、ステップが 120 分以内に完了しない場合は、スーパーバイザーに通知されます。ステップ 4 の先行ジョブはステップ 3 の後続ジョブなので、ステップ 4 は、タイムカード・ファイルを処理し、タイムカード報告書を印刷するジョブが完了するまで待機します。

ジョブの完了後に、タイムカード報告書が検査されたことがクラークに通知されます。実行のフラグが立てられている後続ジョブはありません。タイムカード報告書の検査後に、クラークは手動でステップ 5 を完了します。給与計算を処理し、明細を生成する後続ジョブに実行のフラグを立てます。

ステップ 6 の先行ジョブはステップ 5 の後続ジョブなので、ステップ 6 は、給与計算を処理し、明細を生成するジョブが完了するまで待機します。ジョブの完了後に、クラークおよびスーパーバイザーに給与計算が完了したことが通知されます。これで明細は印刷して配布することができます。

Work Flow Manager の詳細については、オンライン・ヘルプを参照してください。

新規ワークフローの作成:

新規ワークフローの作成時に、ワークフローの開始方法、その最大処理時間、タスク・ステップおよびその実行順序、スケジューリング、通知および文書の詳細を指定する必要があります。

新規ワークフローを作成するには、以下のステップを実行する必要があります。

1. System i ナビゲーターで、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**実行管理機能**」 → 「**Advanced Job Scheduler**」の順に展開します。
2. 「**ワークフロー・マネージャー (Work Flow Manager)**」を右クリックし、「**新規ワークフロー (New Work Flow)**」を選択します。「新規ワークフロー (New Work Flow)」ウィンドウが表示されます。

「新規ワークフロー (New Work Flow)」ウィンドウを完成させるための詳細については、オンライン・ヘルプを参照してください。

ワークフローを一度セットアップすると、ワークフロー名を右マウス・ボタン・クリックし、「**ワークフロー状況 (Work Flow Status)**」をクリックすることで、ワークフローを管理できます。

ワークフローの開始:

ワークフローの開始時に、ワークフローを最初の順序で開始するか、それとも特定の順序で開始するかを選択できます。

ワークフローを開始するには、以下のステップに従います。

1. System i ナビゲーターで、「**実行管理機能**」 → 「**Advanced Job Scheduler**」 → 「**ワークフロー・マネージャー (Work Flow Manager)**」の順に展開し、ワークフローを右クリックして、「**開始 (Start)**」を選択します。 「ワークフローの開始 (Start Work Flow)」ウィンドウが表示されます。
2. ワークフローを最初の順序で開始するか、または特定の順序で開始するかを選択します。最初の順序以外の順序で開始することを選択した場合、その前のすべてのステップは完了済みとマークされます。

「ワークフローの開始 (Start Work Flow)」ウィンドウの詳細については、オンライン・ヘルプを参照してください。

ワークフローの処理:

ワークフローはその実行時に、「ワークフロー状況 (Work Flow Status)」ウィンドウを使用して制御およびモニターできます。

「ワークフロー状況 (Work Flow Status)」ウィンドウには、**「ユーザー接続」 → *ご使用のシステム* → 「実行管理機能」 → 「Advanced Job Scheduler」 → 「ワークフロー・マネージャー (Work Flow Manager)」**の順に展開してアクセスします。ワークフローを右クリックし、**「状況 (Status)」**を選択します。

- 「汎用 (General)」ウィンドウに、ワークフローの現在の状況が表示されます。
- 「ステップ (Steps)」ウィンドウには、現在ワークフローに定義されているすべてのステップのリストが示されます。

  ステップが自動または手動のいずれとして定義されているか、およびステップがいつ開始および終了したかを確認できます。
  - 手動ステップに完了済みとしてマークするには、正しいステップを選択して、**「完了 (Complete)」**ボックスにチェックを付けます。
  - ステップのすべての先行 Advanced Job Scheduler ジョブが完了していれば、手動ステップは任意の順序で完了済みとしてマークすることができます。
  - リスト内で完了している自動ステップがそれ以上ない場合、手動ステップは未完了としてマークすることができます。
  - ワークフローは、任意のステップで手動で開始できます。これにより以前のすべてのステップはバイパスされます。

  リストを最新表示するには、**「最新表示 (Refresh)」**をクリックします。
- 「文書 (Documentation)」ウィンドウは、ワークフローの文書テキストを表示します。

Advanced Job Scheduler のジョブ・アクティビティーのモニター:

Advanced Job Scheduler を使用して、ジョブまたはジョブ・グループの履歴または状況を表示することができます。アクティビティー保持をセットアップすることもできます。アクティビティー保持は、ジョブのアクティビティー記録を保持する期間を表します。

スケジュール済みジョブのアクティビティー:

スケジュール済みジョブのアクティビティーでは、Advanced Job Scheduler アクティビティーの記録を保持する期間を指定できます。指定できる値は、1 から 999 までの日数または出現数です。アクティビティーを特定の日数のあいだ保持するか、またはジョブごとに特定の出現数のあいだアクティビティーを保持するかを指定できます。

スケジュール済みジョブについて、以下の詳細が表示されます。

- 名前。スケジュール済みジョブの名前。
- グループ。ジョブのジョブ・グループの名前。
- 順序。グループ内のジョブの順序番号 (ジョブがジョブ・グループに含まれている場合)。
- 完了状況。ジョブの状況。
- 開始済み。ジョブがいつ実行を開始したか。
- 終了。ジョブがいつ終了したか。
- 経過時間。ジョブの処理に要した時間および分。

アクティビティー保持の指定:

このステップでは、アクティビティー保存を指定する方法を示します。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**スケジュール済みジョブ・アクティビティー (Scheduled Job Activity)**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。

スケジュール済みジョブ・アクティビティーの詳細の表示:

このステップでは、スケジュール済みジョブ・アクティビティーの詳細を表示する方法を示します。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**スケジュール済みジョブ・アクティビティー (Scheduled Job Activity)**」をダブルクリックします。

特定のジョブのスケジュール済みジョブ・アクティビティーの表示:

このステップでは、特定のジョブのスケジュール済みジョブ・アクティビティーを表示する方法を示します。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**スケジュール済みジョブ (Scheduled jobs)**」をクリックします。
4. アクティビティーを表示する「**ジョブ名 (Job Name)**」を右マウス・ボタン・クリックし、「**アクティビティー (Activity)**」をクリックします。

アクティビティー・ログの詳細の表示:

アクティビティー・ログでは、ジョブの追加、変更、または投入などスケジューラー内のアクティビティーを表示します。セキュリティー違反、スケジュール済みジョブによって処理される順序、および受け取ったすべてのエラーが表示されます。以前のアクティビティーの日時も表示されます。

詳細なメッセージ情報を表示するには、日時をダブルクリックします。アクティビティー・ログの詳細を表示するには、以下のようにします。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**アクティビティー・ログ (Activity Log)**」をクリックします。今日の項目が表示されます。この選択基準を変更するには、「オプション (Options)」メニューから「**組み込み (Include)**」を選択します。

特定のジョブのアクティビティー・ログの表示:

このステップでは、特定のジョブのアクティビティー・ログを表示する方法を示します。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を展開します。
3. 「**スケジュール済みジョブ (Scheduled jobs)**」をクリックします。
4. アクティビティー・ログを表示する「**ジョブ名 (Job Name)**」を右マウス・ボタン・クリックし、「**アクティビティー・ログ (Activity log)**」をクリックします。

ジョブのプロパティーの「**最終実行 (Last Run)**」ページを使用して、ジョブの進行状況を表示することもできます。 CL プログラム内のあるステップの前または後に「ジョブ・スケジューラーを使用したステッ

プの設定 (SETSTPJS)」コマンドを、ジョブの進行状況を示す記述とともに指定します。 ジョブがプログラム内で SETSTPJS コマンドに達すると、 関連付けられた記述が「最終実行 (Last Run)」ページおよびワイヤレス装置に表示されます。

Advanced Job Scheduler を使用したメッセージのモニター:

ジョブのコマンド・リスト内の各コマンドはメッセージ ID を取ることができ、その ID はモニターに使用されます。 ジョブを実行して、選択したコマンドに入力したメッセージのいずれかに一致するエラー・メッセージが発行されると、ジョブはエラーのログを記録しつつ、リストの次のコマンドの処理を継続します。

右端から 2 桁または 4 桁にゼロを指定した場合は (たとえば ppmm00)、総称メッセージ ID が指定されたことになります。たとえば、CPF0000 を指定した場合は、すべての CPF メッセージがモニターされます。

コマンドにメッセージ ID を追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックします。
3. 「**スケジュール済みジョブ (Scheduled Jobs)**」をクリックして、ジョブをリストします。
4. スケジュール済みジョブを右マウス・ボタンでクリックして、「**プロパティー (Properties)**」をクリックします。
5. リストからコマンドを選択して、「**プロパティー (Properties)**」をクリックします。
6. 「**メッセージ (Messages)**」をクリックします。
7. モニターするメッセージ ID を入力して、「**追加 (Add)**」をクリックします。

ローカル・データ域の作成および処理:

ローカル・データ域とは、ジョブ用に割り振られたスペース部分のことです。すべてのジョブがローカル・データ域を使用するわけではありませんが、一部のジョブは使用します。ジョブ内の各コマンドは、そのジョブのローカル・データ域へのアクセスを持っています。以前に手作業で追加パラメーターを指定する必要があったジョブをスケジュールしている場合、ローカル・データ域を使用したいと思うことがあります。追加パラメーターを指定する際にローカル・データ域を使用すれば、ジョブを開始するたびにパラメーターを手作業で指定する必要はなくなります。

スケジュール済みジョブにローカル・データ域情報を指定するには、 以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」 → 「**スケジュール済みジョブ (Scheduled Jobs)**」の順に展開します。
3. ジョブを右マウス・ボタンでクリックし、「**プロパティー (Properties)**」をクリックします。
4. 必要に応じて、「ローカル・データ域 (Local Data Area)」ウィンドウを編集します。

ローカル・データ域の詳細を入力するにあたって、詳しい情報はオンライン・ヘルプを参照してください。

アプリケーション制御とジョブ制御の作成および処理:

アプリケーションとは、処理のために複数のジョブが 1 つにまとめられたものを指します。アプリケーションはジョブ・グループよりも範囲が広く、必ずしも逐次処理されるとは限りません。アプリケーションでは複数のジョブが同時に処理され、ジョブが次に処理されるまで待機する必要はありません。同じアプリケーション内のすべてのジョブは連携して処理することができ、またそれぞれのジョブが独自のジョブ・デフ

ォルトのセットを持つこともできます。ジョブ制御は、ジョブ・スケジューラーにジョブを追加した時点でそのジョブに割り当てられるデフォルトであり、またそのジョブを投入した時点で使用されるデフォルト値です。

アプリケーションとは、処理のために複数のジョブが 1 つにまとめられたものを指します。たとえば、給与計算に使用する一連のジョブがあり、それをまとめて 1 つの会計処理にするという場合がそうです。

ジョブ制御は、ジョブ・スケジューラーにジョブを追加した時点でそのジョブに割り当てられるデフォルトであり、またそのジョブを投入した時点で使用されるデフォルト値です。ジョブ制御デフォルト値には、カレンダー、祝祭日カレンダー、ジョブ待ち行列、ジョブ記述などの内容が含まれます。

システムにある既存のアプリケーション制御またはジョブ制御はすべて表示することができます。アプリケーション制御またはジョブ制御は、新しいものを追加したり、既存のものに基づいて新しいものを追加したり、除去したりすることができます。また、アプリケーションまたはジョブ制御を選択して、そのプロパティーを表示して変更することもできます。

新規アプリケーション/ジョブ制御を作成するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**アプリケーション/ジョブ制御 (Applications/Job Controls)**」タブをクリックします。
4. 「**新規 (New)**」をクリックし、アプリケーションの名前を入力します。
5. アプリケーションの説明を入力します。
6. アプリケーションの連絡先を選択します。連絡先とは、アプリケーション内のジョブに問題が起きた場合に連絡を取るユーザーの名前です。 1 つのアプリケーションに 5 つまで指定できます。連絡先リストの連絡先を追加または除去することもできます。
7. このアプリケーションを識別するのに役立つ追加情報を入力することができます。入力された情報はこの新規アプリケーションに関連づけられます。問題が起きたときに、この情報が役に立つことがあります。

通知を扱う作業:

通知内では、一連のタスクを実行することができます。通知を使用することにより、受信者プロパティーと報告書配布リスト・プロパティーを指定できます。また、E メールを送信したり、指定した時間内に受信者からの応答がない場合はエスカレーション・リストをセットアップすることもできます。

E メールを送信できるようになるには、通知のために使用するメール・サーバーの指定を行うことが必要です。

Advanced Job Scheduler の通知機能の特長は以下のとおりです。

受信者

ジョブをスケジュールする際には、指定した受信者に通知メッセージを送信するかどうかを指定できます。ジョブが失敗したとき、ジョブが正常に完了したとき、または指定した時間制限内にジョブが開始しなかったときに、通知メッセージを送信することができます。指定した受信者ごとに、受信者のプロパティーを定義することが必要です。受信者のプロパティーにアクセスするには、「**Advanced Job Scheduler**」 → 「**通知 (Notification)**」 → 「**受信者 (Recipients)**」の順に選択し、受信者のリストから受信者を選択します。

報告書配布リスト

配布に適格なスプール・ファイルのリストを指定するために、報告書配布リストを使用します。ジョブからスプール・ファイルが作成されるたびに、スプール・ファイル・リストにあるものと一致するかどうかがチェックされます。一致すると、そのスプール・ファイルに関連付けられている受信者は、スプール・ファイルのコピーを E メールで受け取るか、 またはスプール・ファイルの複製を受信者の出力待ち行列に受け取ります (あるいはその両方)。報告書配布リストにアクセスするには、「**Advanced Job Scheduler**」 → 「**通知 (Notification)**」 → 「**報告書配布リスト (Report distribution list)**」の順に選択します。

E メール

受信者のリストに定義されている受信者および特定の E メール・アドレスに E メール・メッセージを送信できます。受信者のプロパティーには、メッセージの送信先となる E メール・アドレスを指定しておくことが必要です。 E メール・メッセージを送信するときには、スプール・ファイルを添付します。スプール・ファイルは PDF 形式で送信できます。また、指定した期間に指定した受信者からの応答がない場合はエスカレーション・リストをセットアップすることもできます。

E メールに添付するスプール・ファイルの指定:

E メールにスプール・ファイルを添付するには、以下のようにします。

1. 「System i ナビゲーター」ウィンドウから「**基本操作 (Basic Operations)**」を展開します。
2. 「**プリンター出力 (Printer Output)**」をクリックします。
3. スプール・ファイルを右マウス・ボタン・クリックして、「**AJS 経由で送信 (Send via AJS)**」をクリックします。
4. 受信者、件名、およびメッセージを指定します。

注: 上記の操作は「**出力待ち行列 (Output Queues)**」からも実行できます。

エスカレーション・リスト

エスカレーション・リストは受信者のリストを降順で指定します。受信者には、リストされている順序で通知が送られます。最初の受信者がメッセージに応答しない場合に、メッセージが次の受信者に送信されます。応答されるまで、この処理が繰り返されます。エスカレーション・リストを定義するには、「**Advanced Job Scheduler**」 → 「**通知 (Notification)**」 → 「**エスカレーション・リスト (Escalation Lists)**」の順に進みます。

メッセージがエスカレートしないようにする:

メッセージがエスカレートしないようにするには、以下のようにします。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」 → 「**通知 (Notification)**」 → 「**E メール (E-mail)**」 → 「**送信済み (Sent)**」の順にクリックします。
3. エスカレート中のメッセージを右マウス・ボタン・クリックして、「**停止 (Stop)**」をクリックします。

   注: エスカレート中のメッセージだけを表示するには、「System i ナビゲーター」ウィンドウから「**表示 (View)**」 → 「**このビューのカスタマイズ (Customize this view)**」 → 「**組み込み (Include)**」 を選択します。「**タイプ (Type)**」フィールドで、「**エスカレート中 (Escalating)**」を選択します。

ライブラリー・リストの処理:

ライブラリー・リストはライブラリーのユーザー定義リストで、ジョブを処理する際に Advanced Job Scheduler によって使用されます。

ライブラリー・リストとは、ライブラリーのユーザー定義リストのことで、Advanced Job Scheduler のジョブにより処理中に必要な情報を検索するときに使用されます。複数のライブラリー・リストの表示、新規ライブラリー・リストの追加、既存のライブラリー・リストに基づいた新規ライブラリー・リストの追加、および現在スケジュールされているジョブで使用されていない既存のライブラリー・リストの除去を行うことができます。

また、リストを選択し、そのプロパティーを表示して変更することもできます。ライブラリー・リストにはライブラリーを 250 個まで入れることができます。

新規ライブラリー・リストを追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**ライブラリー・リスト (Library Lists)**」タブをクリックします。
4. 「**新規 (New)**」をクリックし、ライブラリー・リストの名前を入力します。
5. ライブラリー・リストの説明を入力します。
6. 「**参照 (Browse)**」をクリックして既存のライブラリーのリストを表示し、ライブラリーをクリックします。
7. 「**追加 (Add)**」をクリックして、選択したライブラリーのリストを追加します。

コマンド変数の処理:

コマンド変数 (以前はパラメーターと呼ばれていた) とは、保管しておいて、Advanced Job Scheduler を通してサブミットされるジョブ内で使用する変数のことです。コマンド変数の例としては、各月の始まり、部門番号、会社番号などがあります。

コマンド変数 (以前はパラメーターと呼ばれていた) とは、Advanced Job Scheduler に保管しておいて、Advanced Job Scheduler を通して投入されるジョブ内で使用する変数のことです。コマンド変数には、スケジュールされたジョブのコマンド・ストリング内部で置き換えられる情報が入ります。 コマンド変数の例としては、各月の始まり、会社の部門番号、会社番号などがあります。複数のコマンド変数の表示、新規コマンド変数の追加、既存のコマンド変数に基づいた新規コマンド変数の追加、および現在スケジュールされているジョブで使用されていないコマンド変数の除去を行うことができます。

既存のコマンド変数を選択し、そのプロパティーを表示して変更することができます。

新規コマンド変数を追加するには、以下の手順で行います。

1. 「System i ナビゲーター」ウィンドウから「**実行管理機能**」を展開します。
2. 「**Advanced Job Scheduler**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
3. 「**コマンド変数 (Command Variables)**」タブをクリックします。
4. 「**新規 (New)**」をクリックし、コマンド変数の名前を入力します。
5. コマンド変数の説明を入力します。
6. コマンド変数の長さを入力します。長さに指定できる範囲は 1 から 90 までです。
7. 置換値を提供する方法を選択します。
   a. コマンド変数に使用するデータを指定します。このフィールドには任意の文字を使用します。データの文字数は、「長さ (Length)」フィールドに指定されている長さより長くすることはできません。
   b. 日付を計算する数式を入力します。 (例については、オンライン・ヘルプを参照してください。)

c. 置換値を検索するために使用するプログラム名を入力します。
d. 置換値を検索するために使用するライブラリーを入力します。
e. 実行時にシステム・オペレーターから置換値が検索されるようにするかどうかを選択します。

Advanced Job Scheduler for Wireless による作業:

Advanced Job Scheduler for Wireless は次の 2 種類の装置で動作します。 1 つは Wireless Markup Language (WML) 装置であるインターネット携帯電話です。もう 1 つは Hypertext Markup Language (HTML) を使用する PDA または PC Web ブラウザーです。このトピックでは、これら 2 つの装置を略して、それぞれ WML と HTML で表します。

ハードウェアおよびソフトウェア要件:

Advanced Job Scheduler for Wireless を実行する前に、必要なソフトウェアおよびハードウェア要件をすべて満たしていることを確認します。

Advanced Job Scheduler for Wireless を実行するには、以下のものが必要です。

- Advanced Job Scheduler (5761-JS1) ライセンス・プログラム: Advanced Job Scheduler for Wireless が組み込まれた Advanced Job Scheduler 製品。
- 機能を実行するための装置
  - ワイヤレス・インターネット・サービスの機能を備えたインターネット電話
  - Web ブラウザー、ワイヤレス・モデム、およびワイヤレス・インターネット・サービスの機能を備えた PDA
  - ワークステーション上の従来の Web ブラウザー
- i5/OS V5R3 以降を実行する TCP/IP ネットワーク内のシステム。
- セントラル・システム上で実行する Web アプリケーション・サーバー。たとえば、以下のものがあります。
  - ASF Jakarta Tomcat Application サーバー
  - セントラル・システム上で実行する、サーブレットをホストする機能を備えたそれ以外のアプリケーション・サーバー
- システム上にインストールされた HTTP サーバー
- 使用する HTTP サーバーが Advanced Job Scheduler ワイヤレス機能を備えていることを識別します。そのためには、Advanced Job Scheduler がインストールされたシステムに文字ベース・インターフェースを使用して接続します。 その後、次のコマンドを指定します。

```
CALL QIJS/QIJSCINT
```

装置の選択:

インターネット電話とワイヤレス PDA はともに急速な変化を遂げつつある技術です。 とはいえ、インターネット電話とワイヤレス PDA は、画面サイズ、ユーザー・インターフェース、その他多くの重要な特性の点で異なっています。このトピックでは、Advanced Job Scheduler for Wireless と互換性のある装置を選ぶのに役立つ情報を提供します。その他のワイヤレス装置も、ワイヤレス・インターネット・ブラウズをサポートしていれば互換性がありますが、対話が異なっている場合があります。

「**Internet-ready telephones (インターネット電話)**」: Advanced Job Scheduler for Wireless の使用に、インターネット電話を選択できます。

「**PDA**」: Advanced Job Scheduler for Wireless の使用に、PDA を選択できます。

「**PC**」: 従来の Web ブラウザーで Advanced Job Scheduler for Wireless を使用することもできます。

ワイヤレス環境の構成:

Advanced Job Scheduler for Wireless が適切に実行されるようにするには、Web アプリケーション・サーバーおよびファイアウォール構成を変更する必要があります。

Advanced Job Scheduler for Wireless の使用を開始する前に、以下の構成またはセットアップが適正に行われているか確認してください。

1. Web アプリケーション・サーバーの構成。ASF Jakarta Tomcat サーブレット・エンジンを使用して実行するように、Advanced Job Scheduler for Wireless をセットアップします。構成に関するこの説明では、Web アプリケーション・サーバーを作成および開始する方法について明記されています。また、Advanced Job Server のワイヤレス機能を使用して作業を開始する前に実行しておく必要のあるプログラムを指定します。
2. ファイアウォールの構成。System i ナビゲーター (ワイヤレス対応) を使用するときに、インターネットからシステムにアクセスします。ファイアウォールがある場合、System i ナビゲーター (ワイヤレス対応) を実行するようにファイアウォール・セットアップを変更しなければならない場合があります。
3. 言語の選択。デフォルトの言語は英語に設定されていますが、選択した言語で表示するように装置を構成することもできます。

以上のステップが完了したら、サーバーへの接続と Advanced Job Scheduler for Wireless の使用を開始する準備ができたことになります。

Web アプリケーション・サーバーの構成:

Advanced Job Scheduler for Wireless を使用した作業を始める前に、Web アプリケーション・サーバーを開始および構成することが必要です。以下の手順を実行して、ASF Tomcat サーブレット・エンジンをセットアップし、HTTP Server (Apache で稼働する) で Advanced Job Scheduler for Wireless を実行できるようにします。

要件

始める前に、QSECOFR 権限が必要であり、IBM HTTP Server for i5/OS (5761-DG1) ライセンス・プログラムをインストールしておく必要があります。

注: 以下の指示を実行すると、HTTP Server の新規インスタンスが作成されます。既存の HTTP Server に Advanced Job Scheduler をセットアップする場合は、以下の指示を実行することができません。

HTTP Server 上での Advanced Job Scheduler for Wireless の初期設定

次のコマンドを実行することにより、Advanced Job Scheduler for Wireless サーブレットが The Apache Software Foundation Jakarta Tomcat サーブレット・エンジンに追加されます。 同時に、IBM HTTP Server (Apache で稼働する) が Advanced Job SchedulerP という名前でセットアップされ、ポート 8210 を使って要求を listen するようになります。

Advanced Job Scheduler for Wireless を使用した作業を始める前に、システム上の HTTP サーバー・インスタンスで Advanced Job Scheduler for Wireless を初期設定することが必要です。そのためには、次のコマンドを文字ベース・インターフェースで指定します。

```
CALL QIJS/QIJSCINT
```

このコマンドは、システムに提供されているプログラムを実行します。

Web アプリケーション・サーバーを構成し、そこで Advanced Job Scheduler インスタンスの初期設定を終えたら、続けて Advanced Job Scheduler ワイヤレス環境の構成に進むことができます。

言語の選択:

Advanced Job Scheduler for Wireless に接続すると、どの言語を使用するか選択することができます。特に言語を指定しない場合は、システムへの接続に進むことができます。

言語を指定するには、次の URL を使用します。

host. *domain*: *port*/servlet/AJSPervasive?lng= *lang*

- *host*: 製品を含むシステムのホスト名。
- *domain*: ホストが置かれているドメイン。
- *port*: Web サーバーのインスタンスが listen するポート。
- *lang*: 言語を表す 2 文字の ID。使用可能な言語とその 2 文字の ID のリストを、次に示します。 (ar: アラビア語。de: ドイツ語。en: 英語。es: スペイン語。fr: フランス語。it: イタリア語。ja: 日本語)

これで、Advanced Job Scheduler for Wireless を使って作業を始めることができます。

i5/OS オペレーティング・システムへの接続:

ワイヤレス装置を使用して、Advanced Job Scheduler 製品を含むシステムに接続できます。

Advanced Job Scheduler for Wireless の使用を開始するには、まずシステムの URL をワイヤレス装置に指定します。 ワイヤレス装置にシステムの URLを指定するときには、次のフォーマットを使用します。 URL (/servlet/Advanced Job SchedulerPervasive) の末尾を、次のように正確に入力します。

host. *domain*: *port*/servlet/Advanced Job SchedulerPervasive

host: System i のホスト名。 *domain*: システムが置かれているドメイン。 *port*: Web サーバーのインスタンスが listen するポート。デフォルトは 8210 です。

使用する言語を指定する場合は、 言語の選択を参照してください。

インターネット電話および PDA ブラウザーのレイアウト

システムの Advanced Job Scheduler for Wireless 機能への接続が正常に完了すると、初期表示画面にインターネット電話または PDA についての要約情報が表示されます。 要約情報には、情報がどれほど最新か、スケジュール済みジョブの数、存在しているアクティビティー項目の数、およびジョブ・モニターの状況をチェックしたり受信者にメッセージを送信したりするためのオプションが明示されています。また、この要約情報画面の上部には、全体の状況を表す「OK」または「Attention」が表示されます。「Attention」が表示されている場合は、ジョブに注意すべき重要なメッセージがあることを示します。注意を要するジョブには、感嘆符が付けられています。

従来のブラウザーのレイアウト

従来のブラウザーのレイアウトは、インターネット電話や PDA の画面と全く同じです。ただし、コンテンツが画面のサイズより小さくなっています。それで、Web ブラウザーのサイズを小さくすることによって、Advanced Job Scheduler for Wireless の Web ブラウザーを開いたまま、他のアプリケーションの作業を行うスペースを空けることができます。また、PC で従来のインターネット・ブラウザーを使用している

場合は、Advanced Job Scheduler メインメニューから「**すべてを表示 (Show all)**」を選択することができます。すると、Web の 1 ページ分により多くのコンテンツを表示できます。

システムへの接続が正常に完了したら、 次に行うことは接続のカスタマイズです。

接続のカスタマイズ:

ワイヤレス装置を使用する場合に、特定の必要にあわせてインターフェースをカスタマイズすることができます。たとえば、特定のジョブだけを表示し、ジョブのグループ名は表示されないようにしたいと思うことがあります。また、スケジュール済みアクティビティーのリストにはアクセスしないようにしたいと思うこともあります。こうした場合には、ワイヤレス装置の「カスタマイズ (Customize)」ページを使って、ジョブをフィルターしたり、表示設定を変更したりすることができます。

PC、PDA、またはインターネット電話のどれを使用しているかにかかわらず、接続は色々な方法でカスタマイズできます。これらの機能を利用するには、 Job Scheduler for i5/OS  Web サイト (英文) を参照してください。

Advanced Job Scheduler for Wireless の管理:

Advanced Job Scheduler を使った作業でワイヤレス装置を使用することができます。

ワイヤレス装置を使用すると、以下の機能が利用できます。

アクティブ、保留、未解決ジョブの表示
: アクティブ、保留、または未解決の状態にある通常のジョブ (Advanced Job Scheduler ジョブ) またはマネージメント・セントラル・ジョブのリストを表示できます。ジョブ・タイプ、名前、時刻でソート表示することにより、さらにジョブをカスタマイズすることも可能です。また、データ・ライブラリーに組み込むジョブとアクティビティーのデータを指定することもできます。

ジョブ依存関係の表示
: 特定のジョブについて、その先行ジョブと後続ジョブを表示することができます。後続ジョブとは、1 つ以上のジョブ (先行ジョブ) に従属して実行するジョブのことです。後続ジョブが他の後続ジョブの先行ジョブになることもあります。

メッセージの表示
: ジョブに処理待ちのメッセージがある場合、ワイヤレス装置を使用してメッセージ・テキストを表示し、メッセージに返信することができます。

ジョブの開始
: ワイヤレス装置を使用して、ジョブを投入できます。ジョブを投入するときに指定できるオプションは、使用するワイヤレス装置によって異なります。

Advanced Job Scheduler アクティビティーの処理
: ワイヤレス装置から Advanced Job Scheduler アクティビティーと対話することができます。各アクティビティーのオプションは、アクティビティー項目の状況に基づいて異なっています。

国際化対応
: Advanced Job Scheduler for Wireless は、ユーザーの System i (TM) Java(TM) 仮想マシンに関連付けられた国別コードと言語コードを使用して、ワイヤレス装置で使用する言語および日時フォーマットを判別します。 Java 仮想マシンのデフォルト値となっているコードを使用したくない場合は、コードを簡単に変更できます。詳しくは、オンライン・ヘルプを参照してください。

特定のタスクの実行に関する詳細については、オンライン・ヘルプを参照してください。

Advanced Job Scheduler のトラブルシューティング:

スケジュールどおりにジョブが実行しない場合に、これらのトラブルシューティング・メソッドは、対処方法を見つける上で役に立ちます。

Advanced Job Scheduler のトラブルシューティングには、まず Job Scheduler for i5/OS Web サイト (英文) の『Frequently Asked Questions』ページを参照してください。よく尋ねられる質問を参照し、特定の機能を Advanced Job Scheduler で実行する方法について確認してください。

また、スケジュールした時刻にジョブが実行しないときに、検討できる項目のリストを以下に示します。

現行修正レベル

最初に確認すべきことは、使用している修正が現行のものかということです。修正を要求する場合は、必ずすべての修正のリストを要求してください。 累積パッケージにはすべての修正は含まれていません。

ジョブ・モニターのチェック

- ジョブ QIJSSCD は QSYSWRK サブシステム内でアクティブでなければなりません。このジョブがアクティブでない場合は、ジョブ・スケジューラー開始 (STRJS) コマンドを処理します。
- ジョブの状況が 10 分以上「実行中 (RUN)」である場合は、ジョブ・モニターがループ状態になっている可能性があります。ジョブ・モニターがループ状態になっている場合は、*IMMED を指定してジョブを終了し、もう一度ジョブを開始します (STRJS)。
- 応答すべきメッセージがある場合は、C (取り消し) で応答します。ジョブ・モニターは 90 秒遅延してから、モニターを再開します。 モニター・ジョブのジョブ・ログを出力します。このログには、エラー・メッセージが含まれます。

Advanced Job Scheduler ログのチェック

ジョブに対して「ジョブ・スケジューラーのログ表示 (DSPLOGJS)」コマンドを実行します。 F18 を押して、リストの末尾に移動します。そこに、ジョブが実行しなかった理由を説明する項目があります。項目の例としては、リソース障害、アクティブまたはジョブ依存関係の状態、または投入エラーなどが挙げられます。

別のジョブへの依存関係

ジョブが別のジョブに従属している場合、「ジョブの処理」画面でオプション 10 を選択してジョブ依存関係を表示します。 F8 を押して、すべての先行ジョブをリストします。すべての先行ジョブの「**完了 (Complete)**」列が *YES になっていないと、従属ジョブは実行できません。

ジョブの進行状況の追跡

ジョブが適切に機能していない場合は、CL プログラム内のあるステップの前または後で「ジョブ・スケジューラーを使用したステップの設定 (SETSTPJS)」コマンドを使用して、問題の判別に役立てることができます。 コマンドを説明テキストとともに CL プログラムに指定します。このコマンドは何回でも必要な回数使用できます。現行コマンドに関連付けられているテキスト記述が、スケジュール済みジョブ・プロパティーの「最終実行 (Last Run)」ページの「コマンド・ステップ (Command step)」フィールドに表示されます。「コマンド・ステップ (Command step)」フィールドは、アクティブ・ジョブの「状況 (Status)」ウィンドウにも表示されます。「コマンド・ステップ (Command step)」フィールドは、ジョブが SETSTPJS コマンドに処理されるたびに自動的に更新されます。 このコマンドは、ジョブの進行状況を判断するのに役立ちます。

以下のデータ・サンプルを収集すると、問題分析に役立ちます。

エラー・メッセージ条件
エラーが起きた場所にしたがって、対話式セッション、モニター・ジョブ、またはスケジュール済みジョブのジョブ・ログを出力します。

ジョブ・スケジュールが正しくない
ジョブに対して DSPJOBJS コマンドに OUTPUT(*PRINT) を指定して処理します。 ジョブの内部でカレンダーが使用されている場合は、カレンダー報告書を出力します。ジョブの内部で祝祭日カレンダーが使用されている場合は、祝祭日カレンダー報告書を出力します。ジョブの内部で会計カレンダーが使用されている場合は、Print キーを押して、各会計カレンダー項目ごとに画面を出力します。

Advanced Job Scheduler ログ
問題となっている期間の Advanced Job Scheduler ログは、必ず出力してください。

ファイル QAIJSMST および QAIJSHST
ライブラリー QUSRIJS にあるファイル QAIJSMST と QAIJSHST は、問題の再現を試行する前にジャーナルしておくことが必要です。また、QUSRIJS ライブラリーについては弊社のサポートが必要である場合があります。

ジョブ・スケジュール項目の処理

System i ナビゲーター の「ジョブのプロパティー」-「ジョブ待ち行列」ウィンドウ以外にも、文字ベース・インターフェースを使用してジョブ・スケジュール項目を直接変更することもできます。 以下に、ジョブ・スケジュール項目を処理するときに使用できる文字ベース・インターフェースの共通タスクのリストを示します。

重要: ジョブ・スケジュール項目処理 (WRKJOBSCDE) を使用して、マネージメント・セントラル・スケジューラーまたは Advanced Job Scheduler を使用してスケジュールされたジョブを変更または削除しないでください。 WRKJOBSCDE を使用してジョブを変更または削除すると、マネージメント・セントラルには変更が通知されません。 タスクが期待通りに実行されず、マネージメント・セントラル・サーバーのジョブ・ログにエラー・メッセージが表示される場合があります。

関連概念

64 ページの『ジョブ・スケジュール項目』
システムにマネージメント・セントラル・スケジューラーまたは Advanced Job Scheduler がない場合、文字ベース・インターフェースからアクセスできるジョブ・スケジュール項目を使用してジョブをスケジュールできます。この方法を使用すると、ジョブを繰り返し、または 1 度だけ実行するようにスケジュールできます。

ジョブ・スケジュール項目の追加:

ジョブ・スケジュール項目追加 (ADDJOBSCDE) コマンドにより、項目をジョブ・スケジュールに追加して、バッチ・ジョブをスケジュールできます。このコマンドを使用して、バッチ・ジョブを一度だけ投入するか、または定期間隔で投入するかをスケジュールできます。

コマンド: ジョブ・スケジュール項目追加 (ADDJOBSCDE)

例: このコマンドは、CLEANUP というジョブを金曜日の 11 p.m. ごとに投入します。このジョブはライブラリー CLNUPLIB 内のジョブ記述 CLNUPJOBD を使用します。システムが金曜日の 11 p.m. に電源遮断されているかまたは制限状態である場合、IPL 時またはシステムが制限状態でなくなったときにジョブは投入されません。

```
ADDJOBSCDE   JOB(CLEANUP)  SCDDATE(*NONE)
             CMD(CALL PGM(CLNUPLIB/CLNUPPGM))
             SCDDAY(*FRI)   SCDTIME('23:00:00')
             FRQ(*WEEKLY)   RCYACN(*NOSBM)
             JOBD(CLNUPLIB/CLNUPJOBD)
```

ジョブ・スケジュール項目の変更:

このコマンドはジョブ・スケジュール内の項目を変更しますが、この項目を使用してすでに投入されたジョブには影響を与えません。ジョブ項目を変更するには、文字ベース・インターフェースを使用します。

ジョブ・スケジュール項目を変更するには、項目を追加するために必要とされるものと同じ権限を持っている必要があります。ただし、個々のオブジェクトに対する権限は、項目のそのパラメーターを変更する場合にのみ検査されます。加えて、*JOBCTL 特殊権限を持っていない場合、ユーザー・プロファイルでジョブ・スケジュール・オブジェクトに追加された項目だけを変更できます。

コマンド: ジョブ・スケジュール項目変更 (CHGJOBSCDE)

例: このコマンドは、ジョブ・スケジュール項目 BACKUP の番号 001584 に対して変更を行い、そのジョブがライブラリー QGPL 内のジョブ待ち行列 QBATCH に投入されるようにします。

```
CHGJOBSCDE   JOB(BACKUP)  ENTRYNBR(001584)  JOBQ(QGPL/QBATCH)
```

例: このコマンドは、バッチ・ジョブのスケジュールを、03 年 12 月 15 日の 11 a.m. および各週のその日と同じ曜日にプログラムを実行するように変更します。

```
CHGJOBSCDE   JOB(EXAMPLE)  ENTRYNBR(*ONLY)  CMD(CALL PGM(A))
             FRQ(*WEEKLY) SCDDATE(121503)  SCDTIME(110000)
```

ジョブ・スケジュール項目の保留:

ジョブ・スケジュール項目保留 (HLDJOBSCDE) コマンドにより、ジョブ・スケジュール内の 1 つの項目、すべての項目、または 1 セットの項目を保留にできます。項目を保留にすると、スケジュールされた時刻にジョブは投入されません。ジョブ・スケジュール項目を保留にするには、文字ベース・インターフェースを使用します。

項目を保留にするには、ジョブ制御 (*JOBCTL) 特殊権限を持っている必要があります。持っていない場合は、ユーザー自身が追加した項目だけしか保留にできません。ジョブ・スケジュール項目を保留にするには、以下のようにします。

- 項目は、ジョブ・スケジュール項目解放 (RLSJOBSCDE) またはジョブ・スケジュール項目処理 (WRKJOBSCDE) コマンドを使用して解放されるまで保留されます。
- ジョブは、項目が保留されている間に投入するようにスケジュールされた日時を経過するとしても、解放された場合は投入されません。ジョブは、投入するようにスケジュールされた将来の日付に投入されます。

コマンド: ジョブ・スケジュール項目保留 (HDLJOBSCDE)

例: 以下の例では、ジョブ・スケジュール項目 CLEANUP を保留にします。

```
HLDJOBSCDE   JOB(CLEANUP)
```

ジョブ・スケジュール項目のリストの印刷:

ジョブ・スケジュール項目のリストを印刷するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ・スケジュール項目処理 (WRKJOBSCDE)

例: 以下の例は、ジョブ・スケジュール項目のリストを印刷します。

```
WRKJOBSCDE OUTPUT(*PRINT)
```

例: 以下は、各ジョブ・スケジュール項目の詳細情報を印刷します。

```
WRKJOBSCDE OUTPUT(*PRINT) PRTFMT(*FULL)
```

ジョブ・スケジュール項目の解放:

ジョブ・スケジュール項目解放 (RLSJOBSCDE) コマンドによって、ジョブ・スケジュール内の項目、すべての項目、または項目のセットを解放することができます。 ジョブ・スケジュール項目を解放する場合には、項目の保留中にジョブの投入がスケジュールされた日付および時刻を過ぎても、ジョブは直ちに投入されません。項目の保留中にスケジュールされた時刻を過ぎると、ジョブが失われたことを示す警告メッセージが送信されます。次いで、ジョブは、投入がスケジュールされた将来の日付で投入されます。ジョブ・スケジュール項目を解放するには、文字ベース・インターフェースを使用します。

項目を解放するには、ジョブ制御 (*JOBCTL) 特殊権限が必要です。そうでない場合には、追加した項目だけしか解放することはできません。

コマンド: ジョブ・スケジュール項目解放 (RLSJOBSCDE)

例: この例は、保留状況のすべてのジョブ・スケジュール項目を解放します。

```
RLSJOBSCDE   JOB(*ALL)  ENTRYNBR(*ALL)
```

ジョブ・スケジュール項目の除去:

ジョブ・スケジュール項目除去 (RMVJOBSCDE) コマンドによって、ジョブ・スケジュール内の項目、複数の項目、または総称項目を除去することができます。ジョブ・スケジュール項目は 1 つのバッチ・ジョブと対応していて、ジョブを一度、または定期的なスケジュール間隔で自動的に実行するために必要な情報が入っています。項目が正常に除去されるとメッセージが表示され、同時にそのメッセージはジョブ・スケジュール項目に指定されたメッセージ待ち行列に送られます。ジョブ・スケジュール項目を除去するには、文字ベース・インターフェースを使用します。

項目を除去するには、ジョブ制御 (*JOBCTL) 特殊権限があるユーザー・プロファイルのもとで実行中でなければなりません。そうでない場合には、追加した項目しか除去することはできません。

コマンド: ジョブ・スケジュール項目除去 (RMVJOBSCDE)

例: 以下の例は、ジョブ PAYROLL をジョブ・スケジュールから除去します。

```
RMVJOBSCDE   JOB(PAYROLL)  ENTRYNBR(*ONLY)
```

システム・ジョブが 1 度だけ投入される項目を除去する場合、または項目がジョブ・スケジュール項目除去 (RMVJOBSCDE) コマンドによって除去される場合は、項目に指定されたメッセージ待ち行列にシステム・メッセージ CPC1239 が送られます。 1 度だけ投入される項目のスケジュールされた時刻になってもその項目が保留されていて、その項目の保存属性に *NO が指定されている場合は、その項目はジョブ・スケジュール項目除去コマンドで解放されるときに除去されます。この場合は、メッセージ CPC1245 が項目に指定されたメッセージ待ち行列に送られます。

サブシステムの管理

ジョブはサブシステム上で実行されるため、ジョブの実行機能に影響する潜在的な問題について、サブシステムのアクティビティーをモニターする必要があります。

サブシステムは、システムでのジョブの作業場所です。すべてのユーザー作業はサブシステム内のジョブを実行することによって行われるため、作業パフォーマンスが低下した場合は、この領域をモニターすることが重要です。 System i ナビゲーター で、サブシステムに関係したジョブとジョブ待ち行列を表示できます。 さらに、他のどの領域からでも、ジョブとジョブ待ち行列を表示して、ジョブとジョブ待ち行列に対して同じ機能性を持つことになります。

共通サブシステム・タスク

ここでは、サブシステム上で実行できる最も一般的なタスクについて説明します。

関連概念

11 ページの『サブシステム』
サブシステムとは、作業が処理されるシステム上の場所のことです。サブシステムとは、システムがワークフローとリソース使用を調整するために使用する単一の事前定義された操作環境のことをいいます。システムには、それぞれ独立して作動する複数のサブシステムを含めることができます。サブシステムはリソースを管理します。

関連情報

経験報告: サブシステム構成

サブシステム属性の表示:

サブシステムには属性があります。これらの属性は、サブシステムの現在の状況に関する情報、つまりサブシステム記述に指定されている値に関する情報を提供します。

System i ナビゲーターを使用すると、アクティブ・サブシステムに関する以下の属性を表示できます。

- **サブシステム**: サブシステムの名前、およびサブシステム記述を含むライブラリー。
- **説明**: サブシステムの説明。
- **状況**: サブシステムの現在の状況。ヘルプには、起こり得る状況の詳細が含まれています。
- **アクティブ・ジョブ**: サブシステム内で実行中または実行待機中である現在アクティブ状態にあるジョブの数。この数にはサブシステム・ジョブは含まれません。
- **最大アクティブ・ジョブ数**: サブシステム内で実行中または実行待機中のアクティブ状態になり得るジョブの最大数。
- **サブシステム・ジョブ**: サブシステム・ジョブの名前 (ユーザーおよび番号を含む)。

System i ナビゲーター:

サブシステムの属性を表示するには、以下の手順で行います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**サブシステム**」 → 「**アクティブ・サブシステム**」の順に展開します。
2. 表示するサブシステムを右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。

文字ベース・インターフェース:

文字ベース・インターフェースを使用するには、以下のコマンドを入力します。

コマンド: サブシステム記述表示 (DSPSBSD)

例: このコマンドは、サブシステム QBATCH のサブシステム記述メニューを表示します。

```
DSPSBSD QBATCH
```

サブシステムの停止:

System i ナビゲーター または文字ベース・インターフェースを使用して、1 つまたは複数のアクティブ・サブシステムを停止して、処理されているアクティブ作業をどうするか指定することができます。 サブシステムが停止すると、そのサブシステムでは新しいジョブまたは経路指定ステップは開始されません。

サブシステムが停止する場合に、システムによって処理されていたアクティブ作業をどうするかを指定できます。たとえば、サブシステム内のすべてのジョブはすぐに終了する (即時) ように指定するか、サブシステムが終了する前にジョブは処理を終了できる (制御された) ように指定できます。

重要: 可能な場合はいつでも「制御された終了」オプションを使用して、サブシステムを停止することをお勧めします。こうすることによって、アクティブ・ジョブがそれ自身で終了できます。このオプションを使用すると、サブシステムの終了の前にジョブが確実に終了するようにできます。こうすると、実行中のプログラムは終結処置 (ジョブ終了処理) を行えます。「即時」を指定すると、望まない結果 (データが部分的に更新されるなど) を招く場合があります。

2 つのタイプの停止があります。

制御された方法 (推奨)
サブシステムを制御された方法で終了します。ジョブも制御された方法で終了します。こうすると、実行中のプログラムは終結処置 (ジョブ終了処理) を行えます。終了するジョブが、非同期信号 SIGTERM 用の信号処理プロシージャーを有する場合、そのジョブに対して SIGTERM 信号が生成されます。アプリケーションには、ジョブの終了前にクリーンアップを完了させるための、DELAY パラメーターに指定された時間があります。

即時 サブシステムを即時に終了させます。ジョブも即時に終了します。終了するジョブが、非同期シグナル SIGTERM 用のシグナル処理プロシージャーを有する場合、そのジョブに対して SIGTERM シグナルが生成され、QENDJOBLMT システム値が制限時間を指定します。 SIGTERM シグナルの処理以外は、実行中のプログラムはクリーンアップの実行を許可されません。

関連概念

120 ページの『ジョブの終了: 制御』
制御された仕方でジョブを終了すると、ジョブで実行中のプログラムでジョブ終了終結処置を実行することができます。制御された仕方でジョブを終了するための遅延時間を指定できます。遅延時間がジョブ終了前に終わると、ジョブは即時に終了します。

関連タスク

221 ページの『ジョブ・ログの表示方法』
実行管理機能の中でジョブにアクセスできる場所であれば、サブシステム・エリアまたはメモリー・プール・エリアなど、どこからでもジョブ・ログを参照することができます。ジョブ・ログを表示するには、System i ナビゲーターまたは文字ベース・インターフェースを使用できます。

関連情報

ジョブ・システム値: 即時終了の最大時間

System i ナビゲーター:
System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーターで、**「ユーザー接続」** → **「接続 (connection)」** → **「実行管理機能」** → **「サブシステム」** → **「アクティブ・サブシステム」**の順に展開します。
2. 停止したいサブシステム (複数も可) を右マウス・ボタン・クリックして、**「停止」**をクリックします。
3. サブシステムが停止するときに使用されるオプションを指定します。

4. 「**停止**」をクリックします。

文字ベース・インターフェース:
文字ベース・インターフェースを使用するには、以下のコマンドを入力します。

コマンド: サブシステム終了 (ENDSBS)

例: このコマンドは、QBATCH サブシステム内のすべてのアクティブ・ジョブを終了し、サブシステムを終了します。アクティブ・ジョブは、アプリケーション提供のジョブ終了処理を実行するために 60 秒が許可されます。

```
ENDSBS   SBS(QBATCH)  OPTION(*CNTRLD)  DELAY(60)
```

サブシステム終了のパフォーマンスを向上させるには、サブシステム終了オプション (ENDSBSOPT) パラメーターを使用します。 ENDSBSOPT(*NOJOBLOG) を指定した場合、サブシステムは終了しますが、サブシステム内にあったすべてのジョブに対してジョブ・ログが生成されるわけではありません。

ジョブ内で問題が発生したが、*NOJOBLOG が指定されていた場合、その問題はジョブ・ログ内に記録されないので、問題診断が困難または不可能になることがあります。 LOGOUTPUT(*PND) ジョブ属性を使用した場合、ジョブ・ログは保留状態になっており、書き込まれません。ただし、必要な場合にはジョブ・ログを引き続き使用できます。保留ジョブ・ログについて詳しくは、ジョブ・ログの関連トピックを参照してください。

ENDSBSOPT(*CHGPTY *CHGTSL) を指定した場合、このサブシステム内で終了するすべてのジョブについて、実行優先順位およびタイム・スライスが変更されます。 ジョブはプロセッサー・サイクルに関してそれほど激しく競合することなく、他のサブシステム内で実行中のジョブにあまり影響を与えずに終了します。

ENDSBSOPT パラメーター上には、たとえば以下のように 3 つのオプションすべて (*NOJOBLOG、*CHGPTY、および *CHGTSL) を指定できます。

```
ENDSBSOPT(*NOJOBLOG *CHGPTY *CHGTSL)
```

注: サブシステム名に *ALL を指定し、すべてのジョブを QSYSWRK の下で実行させる場合、*CNTRLD を使用してサブシステムが異常終了しないようにする必要があります。

サブシステムの開始:

サブシステム開始 (STRSBS) コマンドは、コマンドに指定されたサブシステム記述を使用してサブシステムを開始します。 サブシステムが開始されると、システムは、サブシステム記述に指定された必要で、しかも使用可能なリソース（記憶域、ワークステーション、およびジョブ待ち行列）を割り振ります。サブシステムを開始するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連概念

21 ページの『サブシステムはどのように開始するか』
サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

System i ナビゲーター:
System i ナビゲーターを使用してサブシステムを開始するには、以下の指示に従います。

1. 「**ユーザー接続**」 → 「**接続 (Connections)**」 → 「**実行管理機能**」の順に展開します。
2. 「**サブシステム**」を右クリックして、「**サブシステムの開始**」をクリックします。
3. 開始するサブシステムの**名前**および**ライブラリー**を指定して、「**OK**」をクリックします。

文字ベース・インターフェース:
コマンド: サブシステム開始 (STRSBS)

例: このコマンドは、QGPL ライブラリーの TELLER サブシステム記述と関連したユーザー・サブシステムを開始します。サブシステム名は TELLER です。

```
STRSBS SBSD(QGPL/TELLER)
```

サブシステム記述の作成

サブシステム記述は、2 とおりの方法で作成することが可能です。既存のサブシステム記述をコピーしたものを変更することができますし、全く新しい記述を作成することもできます。

以下の 2 とおりの方法を使用できます。

1. 文字ベース・インターフェースを使用して既存のサブシステム記述をコピーするには、以下の指示に従います。
   a. 既存のサブシステム記述の複写オブジェクトを作成 (CRTDUPOBJ) します。 (オブジェクト処理 (WRKOBJ) またはプログラム開発管理機能を使用したオブジェクト処理 (WRKOBJPDM) コマンドを使用することもできます。)
   b. 必要とする方法で機能するように、サブシステム記述のコピーを変更します。 たとえば、ジョブ待ち行列項目を、元のサブシステムが使用するジョブ待ち行列を識別するので、除去することが必要になります。 その後、新規サブシステムが使用するパラメーターを指定する新規ジョブ待ち行列項目を作成する必要があります。

      自動開始ジョブ項目、ワークステーション項目、事前開始ジョブ項目、および通信項目を必ず見直し、2 つのサブシステム間に競合がないことを検査します。たとえば、ワークステーション項目が、両方のサブシステムが同じディスプレイ装置を割り振る原因になっていないことを検査します。
2. 全く新しいサブシステム記述を作成するには、文字ベース・インターフェースを使用し、以下の指示に従います。
   a. サブシステム記述作成 (CRTSBSD)。
   b. ジョブ記述作成 (CRTJOBD)。
   c. 事前開始ジョブ項目追加 (ADDPJE) および経路指定項目追加 (ADDRTGE) に対してクラスを作成 (CRTCLS) する。
   d. サブシステム記述に対して実行処理項目の追加を実行する。
      - ワークステーション項目の追加 (ADDWSE)。
      - ジョブ待ち行列項目追加 (ADDJOBQE)。
      - 通信項目追加 (ADDCMNE)。
      - 自動開始ジョブ項目追加 (ADDAJE)。
      - 事前開始ジョブ項目追加 (ADDPJE)。
   e. サブシステム記述に対して経路指定項目追加 (ADDRTGE) を実行する。

関連概念

11 ページの『サブシステム』
サブシステムとは、作業が処理されるシステム上の場所のことです。サブシステムとは、システムがワークフローとリソース使用を調整するために使用する単一の事前定義された操作環境のことをいいます。システムには、それぞれ独立して作動する複数のサブシステムを含めることができます。サブシステムはリソースを管理します。

13 ページの『サブシステム記述』
サブシステム記述は、システムによって制御される操作環境の特性を定義する情報を含んでいるシステム・オブジェクトです。このオブジェクト・タイプのシステム認識 ID は *SBSD です。サブシステム記述は、どれだけの作業が、どこから、どのようにサブシステムに入ってくるか、およびそれらの作業を実行するためにサブシステムがどのリソースを使用するかを定義しています。アクティブ・サブシステムの名前には、サブシステム記述の単純名が使用されます。

関連情報

経験報告: サブシステム構成

自動開始ジョブ項目の追加:

自動開始ジョブ項目の追加には、文字ベース・インターフェースを使用します。自動開始ジョブは、関連づけられているサブシステムが開始するときに自動的に開始します。この種のジョブは、一般的に特定のサブシステムに関連している初期設定作業を実行します。さらに、自動開始ジョブは繰り返し作業を実行したり、同じサブシステム内の他のジョブに集中的なサービス機能を提供したりします。

コマンド: 自動開始ジョブ項目追加 (ADDAJE)

例: この例では、自動開始ジョブ項目をサブシステム ABC の記述に追加します。

```
ADDAJE SBSD(USERLIB/ABC) JOB(START)
       JOBD(USERLIB/STARTJD)
```

注: 変更を有効にするには、アクティブなサブシステムを終了して再始動する必要があります。

関連概念

14 ページの『自動開始ジョブ項目』
自動開始ジョブ項目は、サブシステムの開始とともに開始される自動開始ジョブを示します。サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

通信項目の追加:

それぞれの通信項目では、1 つ以上の通信装置、装置タイプ、またはプログラム開始要求を受け取ったときにジョブを開始するサブシステムのリモート・ロケーションを記述しています。 サブシステムは通信装置を、その装置が現在他のサブシステムまたはジョブに割り振られていない場合は割り振ることができます。現在割り振られている通信装置は、最終的には割り振り解除され、他のサブシステムで使用可能になります。通信項目をサブシステム記述に追加するには、文字ベース・インターフェースを使用します。

コマンド: 通信項目追加 (ADDCMNE)

例: この例では、COMDEV という名前でモードが *ANY の APPC 装置の通信項目を、ライブラリー ALIB にあるサブシステム記述 SBS1 に追加します。 DFTUSR パラメーターはデフォルトでは *NONE になりますが、これは有効なセキュリティー情報がプログラム開始要求に提供されていなければ、この項目によってシステムに投入されるジョブはないことを意味します。

```
ADDCMNE   SBSD(ALIB/SBS1)  DEV(COMDEV)
```

注: DEV パラメーターまたは RMTLOCNAME パラメーターのいずれか (両方ではない) を指定する必要があります。

関連概念

15 ページの『通信項目』
通信作業項目は、処理する通信ジョブのソースをサブシステムに示します。 ジョブ処理は、サブシステムが通信プログラム開始要求をリモート・システムから受け取り、その要求の適切な経路指定項目が検出されたときに開始されます。

ジョブ待ち行列項目の追加:

ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列から開始されるジョブはバッチ・ジョブです。ジョブ待ち行列項目は、文字ベース・インターフェースを使用して追加します。

以下の項目をジョブ待ち行列項目に指定できます。

- ジョブ待ち行列名 (JOBQ)
- ジョブ待ち行列から同時にアクティブにできるジョブの最大数 (MAXACT)
- ジョブの開始元にできるジョブ待ち行列をサブシステムが選択する順序 (SEQNBR)
- 指定されたジョブ待ち行列の優先順位で同時にアクティブにできるジョブの最大数 (MAXPTYn)

コマンド: ジョブ待ち行列項目追加 (ADDJOBQE)

例: このコマンドは、(QGPL ライブラリー内の) NIGHT ジョブ待ち行列のジョブ待ち行列項目を、QGPL ライブラリーに含まれている NIGHTSBS サブシステム記述に追加します。この項目は、NIGHT ジョブ待ち行列から最大で 3 つのバッチ・ジョブを、サブシステム内で同時にアクティブにできることを指定します。デフォルトの順序番号には 10 が想定されます。

```
ADDJOBQE   SBSD(QGPL/NIGHTSBS)  JOBQ(QGPL/NIGHT)  MAXACT(3)
```

関連概念

71 ページの『ジョブ待ち行列項目』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列項目には、ジョブ待ち行列の処理方法を制御する 5 つのパラメーターがあります。

15 ページの『ジョブ待ち行列項目』
サブシステム記述内のジョブ待ち行列項目は、サブシステムがジョブを受け取るジョブ待ち行列を指定します。サブシステムの開始時に、サブシステムはサブシステムのジョブ待ち行列項目内で定義されたそれぞれのジョブ待ち行列の割り振りを試行します。

事前開始ジョブ項目の追加:

事前開始ジョブ項目は、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの入力時に開始できる事前開始ジョブを示します。 事前開始ジョブ項目は、文字ベース・インターフェースを使用してサブシステム記述に追加できます。

コマンド: 事前開始ジョブ項目追加 (ADDPJE)

例: 以下の例では、事前開始ジョブ項目をサブシステム記述 ABC に追加します。

```
ADDPJE SBSD(USERLIB/ABC) PGM(START)
       JOBD(USERLIB/STARTPJ)
```

関連概念

54 ページの『事前開始ジョブ項目』
事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

246 ページの『事前開始ジョブの調査』
このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

経路指定項目の追加:

それぞれの経路指定項目は、ジョブの経路指定ステップを開始するために使用するパラメーターを指定します。経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目をサブシステム記述に追加するには、文字ベース・インターフェースを使用します。

コマンド: 経路指定項目追加 (ADDRTGE)

例: このコマンドは、経路指定項目 46 を ORDLIB ライブラリー内のサブシステム記述 PERT に追加します。経路指定項目 46 を使用するには、経路指定データの先頭が、位置 1 で始まる文字ストリング WRKSTN2 でなければなりません。この項目により、同時にいくつでも経路指定ステップをアクティブにすることができます。ライブラリー ORDLIB 内のプログラム GRAPHIT は、ライブラリー MYLIB 内のクラス AZERO を使用して、記憶域プール 2 で実行します。

```
ADDRTGE   SBSD(ORDLIB/PERT)  SEQNBR(46)  CMPVAL(WRKSTN2)
          PGM(ORDLIB/GRAPHIT)  CLS(MYLIB/AZERO)  MAXACT(*NOMAX)
          POOLID(2)
```

関連概念

17 ページの『経路指定項目』
経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目は、サブシステム記述に保管されます。

ワークステーション項目の追加:

ワークステーション項目は、ユーザーがサインオンするか、または他のサブシステムから対話式ジョブを転送するときに、ジョブを開始する場合に使用されます。以下の項目をワークステーション項目に指定できます。パラメーター名は括弧で囲んで示します。ワークステーション項目の追加には、文字ベース・インターフェースを使用します。

- ワークステーション名またはタイプ (WRKSTN または WRKSTNTYPE)
- ジョブ記述名 (JOBD) またはユーザー・プロファイル内のジョブ記述名
- 項目から同時にアクティブにできるジョブの最大数 (MAXACT)
- サブシステムの開始時、またはジョブ転送 (TFRJOB) コマンドと AT パラメーターとで対話式ジョブがサブシステムに投入される時のいずれかの、ワークステーションの割り振り時。

ワークステーション項目をサブシステム記述に追加するには、文字ベース・インターフェースを使用します。

コマンド: ワークステーション項目追加 (ADDWSE)

例: 以下の例では、ワークステーション項目 DSP12 をサブシステム記述 ABC に追加します。

```
ADDWSE SBSD(USERLIB/ABC) WRKSTN(DSP12)
       JOBD(USERLIB/WSE)
```

関連概念

17 ページの『ワークステーション項目』
対話式ジョブとは、ユーザーがディスプレイ装置にサインオンした時点で開始され、ユーザーがサインオフした時点で終了するジョブのことです。このジョブを実行するために、サブシステムはワークステーション項目またはユーザー・プロファイルに指定されているジョブ記述を検索します。

サインオン・ディスプレイ・ファイルの作成:

サインオン・ディスプレイ・ファイルは、サブシステムに割り振られるワークステーションでサインオン画面を表示するために使用されます。 サインオン・ディスプレイ・ファイルは、サブシステムがアクティブな場合に変更できます。 ただし、新規のサインオン・ディスプレイ・ファイルは、次回のサブシステムの開始時までは使用されません。 サインオン・ディスプレイ・ファイルを作成するには、文字ベース・インターフェースを使用します。

新規のサインオン・ディスプレイ・ファイルは、IBM 提供のサインオン・ディスプレイ・ファイルを開始点として使用して作成できます。 このディスプレイ・ファイルのソースは、ソース物理ファイル QDDSSRC 内のライブラリー QGPL にあります。新規のソース物理ファイルを作成し、IBM 提供のディスプレイ・ファイルを、変更を加える前にその新規のソース物理ファイルにコピーすることを強くお勧めします。こうすると、オリジナルの IBM 提供のソースはその後も使用できます。

考慮事項:

- サインオン・ディスプレイ・ファイル内のフィールドが宣言される順序は変更できません。 それらが画面に表示される位置は変更できます。
- 入力または出力バッファーの合計サイズは変更しないでください。バッファーの順序またはサイズが変更されると、重大な問題が発生する可能性があります。
- サインオン・ディスプレイ・ファイルでは、データ記述仕様 (DDS) ヘルプ機能は使用しないでください。
- MAXDEV パラメーターには必ず 256 を指定してください。
- MENUBAR および PULLDOWN キーワードは、サインオン・ディスプレイ・ファイル記述には指定できません。
- ディスプレイ・ファイルのバッファー長は 318 にする必要があります。それが 318 未満であると、サブシステムは、ライブラリー QSYS 内のデフォルトのサインオン画面 QDSIGNON を使用します。
- 著作権を示す行は削除できません。
- メンバー QDSIGNON は、10 文字のパスワードを使用する IBM 提供のサインオン・ディスプレイ・ファイルです。
- メンバー QDSIGNON2 は、128 文字のパスワードを使用する IBM 提供のサインオン・ディスプレイ・ファイルです。

コマンド: ディスプレイ・ファイル作成 (CRTDSPF)

ディスプレイ・ファイルの UBUFFER という隠しフィールドは、それより小さなフィールドを管理するために変更できます。 UBUFFER は 128 バイト長で、ディスプレイ・ファイル内の最終フィールドと示されています。このフィールドは入出力バッファーとして機能するように変更して、このフィールドに指定されたデータを、対話式ジョブの開始時にアプリケーション・プログラムが使用可能となるようにできます。UBUFFER フィールドは、以下の要件を満たしている場合は、それより小さなフィールドを必要なだけ含めるように変更できます。

- 新規フィールドは、ディスプレイ・ファイル内の他のすべてのフィールドに従っている必要があります。画面上のフィールドの位置は、データ記述仕様 (DDS) 内でそれらが配置されている順序がこの要件を満たしていれば、特に指定はありません。
- 長さは合計で 128 とする必要があります。フィールドの長さが 128 を超える場合、一部のデータは渡されません。
- すべてのフィールドは、入出力フィールド (DDS ソースのタイプ B) または隠しフィールド (DDS ソースのタイプ H) である必要があります。

関連情報

多国語環境の一部としてのロケール

DDS 表示装置ファイル

新規サインオン・ディスプレイの指定:

サブシステムは、サブシステム記述の SGNDSPF パラメーターで指定されたサインオン・ディスプレイ・ファイルを使用して、ユーザー・ワークステーションでサインオン画面を作成します。サインオン・ディスプレイ・ファイルを、デフォルト (QDSIGNON) からユーザーが作成したものに変更するには、文字ベース・インターフェースを使用します。

注: 制御サブシステムを変更する前に、画面が有効であることを検査するために、テスト・バージョンのサブシステムを使用してください。

コマンド: サブシステム記述変更 (CHGSBSD)

新規ディスプレイ・ファイルを SGNDSPF パラメーターに指定します。

例: 以下では、サブシステム QBATCH のサインオン・ディスプレイ・ファイルを、デフォルトから MYSIGNON という新規ファイルに変更します。

```
CHGSBSD SBSD(QSYS/QBATCH) SGNDSPF(MYSIGNON)
```

関連情報

多国語環境の一部としてのロケール

DDS 表示装置ファイル

サブシステム記述の変更

サブシステム記述変更 (CHGSBSD) コマンドは、指定されたサブシステム記述の運用属性を変更します。サブシステムがアクティブである間にサブシステム記述を変更することができます。サブシステム記述を変更するには、文字ベース・インターフェースを使用します。

注: ジョブが中断状態になる可能性があるので、サブシステムがアクティブである間には、*RMV 値を POOLS パラメーター上に指定することはできません。

コマンド: サブシステム記述変更 (CHGSBSD)

例: このコマンドは、サブシステム PAYCTL が使用する記憶域プール 2 の定義を、記憶域サイズ 1500K およびアクティビティー・レベル 3 に変更します。サインオン・ディスプレイ・ファイルは、ディスプレイ・ファイル COMPANYA に変更され、QGPL ライブラリーに置かれます。 このコマンドの発行時にサブシステムがアクティブである場合、COMPANYA は次回のサブシステムの開始時まで使用されません。

```
CHGSBSD  SBSD(QGPL/PAYCTL) POOLS((2 1500 3))
         SGNDSPF(QGPL/COMPANYA)
```

関連概念

13 ページの『サブシステム記述』
サブシステム記述は、システムによって制御される操作環境の特性を定義する情報を含んでいるシステム・オブジェクトです。このオブジェクト・タイプのシステム認識 ID は *SBSD です。サブシステム記述は、どれだけの作業が、どこから、どのようにサブシステムに入ってくるか、およびそれらの作業を実行するためにサブシステムがどのリソースを使用するかを定義しています。アクティブ・サブシステムの名前には、サブシステム記述の単純名が使用されます。

自動開始ジョブ項目の変更:

事前に定義されている自動開始ジョブ項目に対して、別のジョブ記述を指定することができます。自動開始ジョブ項目を変更するには、文字ベース・インターフェースを使用します。

コマンド: 自動開始ジョブ項目変更 (CHGAJE)

例: 以下の例は、サブシステム ABC、ライブラリー USERLIB 内の、START という名前の自動開始ジョブ項目により使用されるジョブ記述を変更します。

```
CHGAJE SBSD(USERLIB/ABC) JOB(START)
       JOBD(USERLIB/NEWJD)
```

注: 変更を有効にするには、アクティブなサブシステムを終了して再始動する必要があります。

関連概念

14 ページの『自動開始ジョブ項目』
自動開始ジョブ項目は、サブシステムの開始とともに開始される自動開始ジョブを示します。サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

通信項目の変更:

既存のサブシステム記述内の既存の通信項目の属性は、文字ベース・インターフェースを使用して変更できます。

- ジョブ記述 (JOBD) またはデフォルト・ユーザー・プロファイル (DFTUSR) パラメーターを変更する場合、通信項目も変更されます。ただし、それらのパラメーターの値は、その時点でアクティブなジョブについては変更されません。
- 最大アクティブ・ジョブ数 (MAXACT) パラメーターの値が、通信項目により、アクティブであるジョブの合計数より少ない数に減らされている場合、新規のプログラム開始要求は処理されません。 アクティブ・ジョブは実行を継続します。しかし、追加のプログラム開始要求は、アクティブ・ジョブの数が MAXACT パラメーターに指定されている値を下回るまで処理されません。

コマンド: 通信項目変更 (CHGCMNE)

例: この例は、デバイス A12 およびモード *ANY の、(サブシステム記述 QGPL/BAKER 内の) 通信項目を変更します。最大アクティビティー・レベルは *NOMAX に変更されますが、これは、通信項目が同時にアクティブにできるプログラム開始要求の数に制限を課さないことを意味します。ただし、サブシステム記述 BAKER 内の MAXJOBS 値は、サブシステム内でアクティブにできるジョブの合計数を制限します。これには、プログラム開始要求で作成されたものも含まれます。特定の経路指定項目を経由して経路指定できるアクティブ・ジョブの数に対して、ユーザーが指定できる制限もあります。経路指定項目に指定された制限は、特定のプール、または特定のプログラムの反復レベルを使用するジョブの数を制御できます。すべての場合に、これらのどの制限も、プログラム開始要求の処理の結果として超過することはできません。

```
CHGCMNE   SBSD(QGPL/BAKER)  DEV(A12)  MAXACT(*NOMAX)
```

関連概念

15 ページの『通信項目』
通信作業項目は、処理する通信ジョブのソースをサブシステムに示します。 ジョブ処理は、サブシステムが通信プログラム開始要求をリモート・システムから受け取り、その要求の適切な経路指定項目が検出されたときに開始されます。

ジョブ待ち行列項目の変更:

指定されたサブシステム記述内の既存のジョブ待ち行列項目は変更可能です。このコマンドは、サブシステムがアクティブでも非アクティブでも発行できます。サブシステム内のジョブ待ち行列項目を変更するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ待ち行列項目変更 (CHGJOBQE)

例: このコマンドは、ライブラリー QGPL 内のジョブ待ち行列 QBATCH から同時にアクティブにできるジョブの最大数を変更します。ジョブ待ち行列項目の順序番号は変更されません。 QBATCH ジョブ待ち行列からの最大で 4 つのジョブを同時にアクティブにできます。最大で 1 つのジョブを、優先順位レベル 1 からアクティブにできます。優先順位レベル 2 から同時にアクティブにできるジョブの最大数には制限はありません。優先順位レベル 3 から 9 は変更されません。

```
CHGJOBQE   SBSD(QGPL/QBATCH)  JOBQ(QGPL/QBATCH)  MAXACT(4)
           MAXPTY1(1)  MAXPTY2(*NOMAX)
```

関連概念

71 ページの『ジョブ待ち行列項目』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列項目には、ジョブ待ち行列の処理方法を制御する 5 つのパラメーターがあります。

15 ページの『ジョブ待ち行列項目』
サブシステム記述内のジョブ待ち行列項目は、サブシステムがジョブを受け取るジョブ待ち行列を指定します。サブシステムの開始時に、サブシステムはサブシステムのジョブ待ち行列項目内で定義されたそれぞれのジョブ待ち行列の割り振りを試行します。

事前開始ジョブ項目の変更:

指定されたサブシステム記述内の事前開始ジョブ項目は変更可能です。事前開始ジョブ項目の変更時に、サブシステムはアクティブであってもかまいません。サブシステムがアクティブであるときに項目に加えられた変更は、時間を経て反映されます。コマンドの発行後に開始されたすべての新規事前開始ジョブは、新規のジョブ関連値を使用します。このコマンドは、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの発行時に開始できる事前開始ジョブを識別します。

サブシステム記述の事前開始ジョブ項目を変更するには、文字ベース・インターフェースを使用します。

コマンド: 事前開始ジョブ項目変更 (CHGPJE)

例: この例は、QGPL ライブラリーに含まれる PJSBS サブシステム記述で、QGPL ライブラリー内の PGM1 プログラムの事前開始ジョブ項目を変更します。この項目に関連付けられた事前開始ジョブは、次回に QGPL ライブラリー内の PJSBS サブシステム記述が開始されるときには開始されません。 STRPJ コマンドは事前開始ジョブを開始するために必要です。 さらにジョブが開始される必要がある場合は、1 つの追加ジョブが開始されます。

```
CHGPJE   SBSD(QGPL/PJSBS)  PGM(QGPL/PGM1)  STRJOBS(*NO)
         THRESHOLD(1)  ADLJOBS(1)
```

関連概念

54 ページの『事前開始ジョブ項目』
事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

246 ページの『事前開始ジョブの調査』
このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

経路指定項目の変更:

指定されたサブシステム記述内の経路指定項目は、文字ベース・インターフェースを使用して変更できます。経路指定項目は、ジョブの経路指定ステップを開始するために使用するパラメーターを指定します。変更を加えるときに、関連サブシステムはアクティブにしておくことができます。

コマンド: 経路指定項目変更 (CHGRTGE)

例: このコマンドは、ライブラリー LIB5 内にあるサブシステム記述 ORDER の経路指定項目 1478 を変更します。同じプログラムが使用されますが、ここではライブラリー LIB6 内のクラス SOFAST を使用して、記憶域プール 3 内で実行します。

```
CHGRTGE   SBSD(LIB5/ORDER)  SEQNBR(1478)  CLS(LIB6/SOFAST)  POOLID(3)
```

関連概念

17 ページの『経路指定項目』
経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目は、サブシステム記述に保管されます。

ワークステーション項目の変更:

文字ベース・インターフェースを使用して、事前に定義されているワークステーション項目に対して、別のジョブ記述を指定することができます。

- ジョブ記述 (JOBD) パラメーターを指定した場合、ワークステーション項目は変更されます。ただし、このパラメーターの値は、この項目から開始されたその時点でアクティブであるジョブについては変更されません。
- 最大アクティブ・ジョブ数 (MAXACT) パラメーターの値が、ワークステーション項目により、アクティブであるワークステーションの合計数より少ない数に減らされている場合、さらに別のワークステーションへのサインオンは許可されません。アクティブ・ワークステーションはサインオフされます。アクティブなワークステーションに対する追加のジョブは、2 次ジョブ移行 (TFRSECJOB) コマンドまたはグループ・ジョブへの移行 (TFRGRPJOB) コマンドにより作成できます。他のワークステーションへのサインオンは、アクティブなワークステーションの数が MAXACT パラメーターに指定された値を下回るまでは許可されません。

コマンド: ワークステーション項目変更 (CHGWSE)

例: このコマンドは、汎用ライブラリーにあるサブシステム BAKER 内のワークステーション A12 のワークステーション項目を変更します。ユーザーのパスワードをサインオン・ディスプレイで入力し、Enter キーを押すと、ワークステーション A12 のジョブが作成されます。

```
CHGWSE   SBSD(QGPL/BAKER)  WRKSTN(A12)  AT(*SIGNON)
```

関連概念

17 ページの『ワークステーション項目』
対話式ジョブとは、ユーザーがディスプレイ装置にサインオンした時点で開始され、ユーザーがサインオフした時点で終了するジョブのことです。このジョブを実行するために、サブシステムはワークステーション項目またはユーザー・プロファイルに指定されているジョブ記述を検索します。

サインオン・ディスプレイの変更:

システムにはデフォルトのサインオン・ディスプレイ・ファイル QDSIGNON が付属しており、これは QSYS ライブラリー内にあります。 マルチリンガル環境をご使用の場合、サインオン画面での表示内容の変更が必要になる場合があります。 あるいは、サインオン画面に企業情報を追加したいという場合もあります。 このような状況では、まず新しいディスプレイ・ファイルを作成することが必要です。これを実行するには、文字ベース・インターフェースを使用します。

サブシステム記述内の SGNDSPF 属性は、サブシステムへのサインオン時にユーザーに表示されるサインオン・ディスプレイ・ファイルを指します。

サインオン画面の変更に使用するステップは、以下のように要約されます。

1. 新しいサインオン・ディスプレイ・ファイルを作成する。
2. システム・デフォルトの代わりにこの変更されたディスプレイ・ファイルを使用するようにサブシステム記述を変更する。
3. 変更をテストする。

関連タスク

186 ページの『サインオン・ディスプレイ・ファイルの作成』
サインオン・ディスプレイ・ファイルは、サブシステムに割り振られるワークステーションでサインオン画面を表示するために使用されます。 サインオン・ディスプレイ・ファイルは、サブシステムがアクティブな場合に変更できます。 ただし、新規のサインオン・ディスプレイ・ファイルは、次回のサブシステムの開始時までは使用されません。 サインオン・ディスプレイ・ファイルを作成するには、文字ベース・インターフェースを使用します。

187 ページの『新規サインオン・ディスプレイの指定』
サブシステムは、サブシステム記述の SGNDSPF パラメーターで指定されたサインオン・ディスプレイ・ファイルを使用して、ユーザー・ワークステーションでサインオン画面を作成します。サインオン・ディスプレイ・ファイルを、デフォルト (QDSIGNON) からユーザーが作成したものに変更するには、文字ベース・インターフェースを使用します。

関連情報

多国語環境の一部としてのロケール

DDS 表示装置ファイル

サブシステム記述の削除

サブシステム記述削除 (DLTSBSD) コマンドは、指定されたサブシステム記述 (すべての実行処理項目またはそれに追加された経路指定項目) をシステムから削除します。ジョブ待ち行列項目追加 (ADDJOBQE) コマンドによってこのサブシステムに割り当てられたジョブ待ち行列は削除されません。 実際には、サブシステム記述 (SBSD) の削除時に、SBSD により参照されるオブジェクトはいずれも削除されません。

関連サブシステムは、削除前に非アクティブにしておく必要があります。サブシステムの削除には、文字ベース・インターフェースを使用します。

コマンド: サブシステム記述削除 (DLTSBSD)

このコマンドは、BAKER と呼ばれる非アクティブ・サブシステム記述をライブラリー LIB1 から削除します。

```
DLTSBSD   SBSD(LIB1/BAKER)
```

自動開始ジョブ項目の除去:

文字ベース・インターフェースを使用して、自動開始ジョブ項目をサブシステム記述から除去できます。

コマンド: 自動開始ジョブ項目除去 (RMVAJE)

例: 以下の例は、ジョブ START の自動開始項目をサブシステム記述 ABC から除去します。

```
RMVAJE SBSD(USERLIB/ABC) JOB(START)
```

注: 変更を有効にするには、アクティブ状態のサブシステムを終了してから再始動する必要があります。

関連概念

14 ページの『自動開始ジョブ項目』
自動開始ジョブ項目は、サブシステムの開始とともに開始される自動開始ジョブを示します。サブシステムの開始時に、システムはいくつかの項目を割り振り、そのサブシステムが作動可能になるのに先立って、自動開始ジョブおよび事前開始ジョブを開始します。

通信項目の除去:

文字ベース・インターフェースを使用して、通信項目をサブシステム記述から除去できます。このコマンドを実行する前に、除去中の通信項目を通じてアクティブ状態のすべてのジョブを終了しなければなりません。

コマンド: 通信項目除去 (RMVCMNE)

例: このコマンドは、装置 COMDEV の通信装置項目をライブラリー LIB2 のサブシステム記述 SBS1 から除去します。

```
RMVCMNE   SBSD(LIB2/SBS1)  DEV(COMDEV)
```

関連概念

15 ページの『通信項目』
通信作業項目は、処理する通信ジョブのソースをサブシステムに示します。 ジョブ処理は、サブシステムが通信プログラム開始要求をリモート・システムから受け取り、その要求の適切な経路指定項目が検出されたときに開始されます。

ジョブ待ち行列項目の除去:

文字ベース・インターフェースを使用して、ジョブ待ち行列項目をサブシステム記述から除去できます。ジョブ待ち行列項目がサブシステム記述から除去される時に、そのジョブ待ち行列上のジョブは待ち行列に残っています。現在アクティブ状態のいずれかのジョブがジョブ待ち行列から開始された場合には、ジョブ待ち行列項目を除去することはできません。

コマンド: ジョブ待ち行列項目除去 (RMVJOBQE)

例: このコマンドは、MYLIB の BATCH2 ジョブ待ち行列を参照するジョブ待ち行列項目をライブラリー MYLIB に保管されている NIGHTRUN サブシステム記述から除去します。

```
RMVJOBQE   SBSD(MYLIB/NIGHTRUN)  JOBQ(MYLIB/BATCH2)
```

関連概念

71 ページの『ジョブ待ち行列項目』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列項目には、ジョブ待ち行列の処理方法を制御する 5 つのパラメーターがあります。

15 ページの『ジョブ待ち行列項目』
サブシステム記述内のジョブ待ち行列項目は、サブシステムがジョブを受け取るジョブ待ち行列を指定します。サブシステムの開始時に、サブシステムはサブシステムのジョブ待ち行列項目内で定義されたそれぞれのジョブ待ち行列の割り振りを試行します。

関連タスク

208 ページの『サブシステムへのジョブ待ち行列の割り当て』
ジョブ待ち行列項目をサブシステム記述に割り当てるには、文字ベース・インターフェースを使用します。

事前開始ジョブ項目の除去:

文字ベース・インターフェースを使用して、事前開始ジョブ項目をサブシステム記述から除去できます。現在アクティブ状態のいずれかのジョブが事前開始ジョブ項目を使用して開始された場合には、その項目を除去することはできません。

ライブラリー名に *LIBL が指定されている項目を除去する時には、指定された名前のプログラムを見つけるために、ライブラリー・リストが検索されます。ライブラリー・リストにプログラムが見つかったが、別のライブラリー名（ライブラリー・リストの後の方にある）をもつ項目が存在している場合には、項目は除去されません。ライブラリー・リストにプログラムが見つからないが、項目が存在している場合は、項目は除去されません。

コマンド: 事前開始ジョブ項目の除去 (RMVPJE)

例: このコマンドは、PGM1 プログラム (QGPLライブラリー中）の事前開始ジョブ項目を QGPL ライブラリーに入っている PJE サブシステム記述から除去します。

```
RMVPJE   SBSD(QGPL/PJE)  PGM(QGPL/PGM1)
```

関連概念

54 ページの『事前開始ジョブ項目』
事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

246 ページの『事前開始ジョブの調査』
このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

関連情報

Experience Report: 事前開始ジョブ項目の調整

経路指定項目の除去:

文字ベース・インターフェースを使用して、指定されたサブシステム記述から経路指定項目を除去できます。サブシステムは、このコマンドの実行時にアクティブ状態にしておくことができます。ただし、経路指定項目を使用して開始された現在アクティブ状態のジョブがある場合は、その項目を除去することはできません。

コマンド: 経路指定項目除去 (RMVRTGE)

例: このコマンドは、経路指定項目 9912 をライブラリー OR のサブシステム記述 PERT から除去します。

```
RMVRTGE   SBSD(OR/PERT)  SEQNBR(9912)
```

関連概念

17 ページの『経路指定項目』
経路指定項目は、使用する主記憶域サブシステム・プール、実行する制御プログラム (一般にはシステム提供のプログラム QCMD)、および追加の実行時情報 (クラス・オブジェクトに保管) を示します。経路指定項目は、サブシステム記述に保管されます。

ワークステーション項目の除去:

文字ベース・インターフェースを使用して、ワークステーション項目をサブシステム記述から除去できます。サブシステムは、このコマンドの実行時にアクティブ状態にしておくことができます。ただし、ワークステーション項目を通じてアクティブ状態のすべてのジョブは、除去する前に終了しなければなりません。

コマンド: ワークステーション項目除去 (RMVWSE)

例: この例は、ワークステーション B53 のワークステーション項目をライブラリー LIB2 の CHARLES という名前のサブシステム記述から除去します。

```
RMVWSE   SBSD(LIB2/CHARLES)  WRKSTN(B53)
```

関連概念

17 ページの『ワークステーション項目』
対話式ジョブとは、ユーザーがディスプレイ装置にサインオンした時点で開始され、ユーザーがサインオフした時点で終了するジョブのことです。このジョブを実行するために、サブシステムはワークステーション項目またはユーザー・プロファイルに指定されているジョブ記述を検索します。

対話式サブシステムの構成

このセクションでは、新しい対話式サブシステムをセットアップする方法について説明します。

これらのステップは、コマンドを手動で入力する場合の方法について説明しています。ただし、制御言語プログラムを使用してサブシステムを作成することにより、回復を目的として構成を簡単に再作成することができます。

新しい対話式サブシステムをセットアップする場合は、そのサブシステムにどれほどの装置を割り振るかを考慮する必要があります。サブシステムは装置管理機能を実行するため (サインオン画面の表示および装置エラー回復の処理など)、1 つのサブシステムに割り振る装置数を制限できます。 詳しくは、Communications limits というトピックを参照してください。

注: このトピックでは、対話式サブシステムの構成に関する概要が説明されています。サブシステムについての経験報告には、各ステップの詳細な説明と各ステップで選択可能な追加のオプションが含まれています。

ライブラリーの作成:

この例は、サブシステム構成オブジェクトを保管するためにライブラリーを作成する方法を示しています。

この例は、SBSLIB をライブラリーとして使用しています。

```
CRTLIB SBSLIB TEXT('LIBRARY TO HOLD SUBSYSTEM CONFIGURATION OBJECTS')
```

クラスの作成:

クラスは、対話式サブシステムの特定のパフォーマンス特性を定義します。 この指示に従ってクラスを作成します。

QINTER クラスと同一のクラスを作成するには、以下のコマンドを入力します。

```
CRTCLS SBSLIB/INTER1 RUNPTY(20) TIMESLICE(2000) PURGE(*YES) DFTWAIT(30)
   TEXT('Custom Interactive Subsystem Class')
```

カスタム対話式サブシステムには QGPL 内の QINTER クラスを使用できます。または単一クラスを作成して、すべての対話式サブシステムに使用できます。またはクラスをそれぞれの対話式サブシステムごとに作成することもできます。

どれを選択するかは、特定のサブシステムのパフォーマンス設定の一部をカスタマイズするかどうかに応じて決まります。 IBM 提供のサブシステムは、それぞれのサブシステムに対して作成されたクラスに付属しており、サブシステム名と同じクラス名を持ちます。

サブシステムと同じ名前でそれぞれのサブシステムのクラスを作成しない場合は、経路指定項目追加 (ADDRTGE) コマンド上でクラス名を指定する必要があります。これは、CLS パラメーターのデフォルトは *SBSD であり、クラス名はサブシステム記述と同じクラス名を持つことを意味するからです。

サブシステム記述の作成:

定義する必要があるサブシステムごとにこのステップを繰り返してサブシステム記述を作成します。

以下は、QINTER の属性と同一の属性でサブシステム記述を作成します。

```
CRTSBSD SBSD(SBSLIB/INTER1) POOLS((1 *BASE) (2 *INTERACT)) SGNDSPF(*QDSIGNON)
```

ジョブ待ち行列の作成:

サブシステム名と同じ名前を使用してサブシステムのジョブ待ち行列を作成し、ジョブ待ち行列項目をサブシステム記述に追加することができます。

このステップは、ジョブ転送 (TFRJOB) コマンドを使用して、ジョブをカスタム・サブシステムに転送する必要がある場合に必要です。

```
CRTJOBQ JOBQ(SBSLIB/INTER1)
ADDJOBQE SBSD(SBSLIB/INTER1) JOBQ(SBSLIB/INTER1) MAXACT(*NOMAX)
```

経路指定項目の追加:

システムに用意されている QINTER の経路指定項目には、追加の機能があります。これらの機能が必要な場合は、カスタマイズしたサブシステム記述にそれらの経路指定項目を追加します。

このステップに従って経路指定項目を追加します。

```
ADDRTGE SBSD(SBSLIB/INTER1) SEQNBR(9999) CMPVAL(*ANY) PGM(QSYS/QCMD) POOLID(2)
```

ワークステーション項目の追加:

サブシステム記述へのワークステーション項目の追加は、サブシステムに装置を割り振る場合の重要なステップです。

どのサブシステムがどの装置を割り振るかを決定する必要があります (AT(*SIGNON))。さらに、あるサブシステムから別のサブシステムに TFRJOB の使用を許可するかどうかを決定します (AT(*ENTER))。

```
ADDWSE SBSD(SBSLIB/PGRM) WRKSTN(PGMR*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/ORDERENT) WRKSTN(ORDERENT*) AT(*SIGNON)
ADDWSE SBSD(QGPL/QINTER) WRKSTN(QPADEV*) AT(*SIGNON)
```

この例では、サブシステムおよび装置の命名規則は、ユーザーが実行する作業のタイプに基づいています。すべてのプログラマーの装置は、PGMR で始まる名前を持ち、PGRM サブシステムで実行されます。すべてのオーダー項目担当者の装置は、ORDERENT で始まる名前を持ち、ORDERENT サブシステムで実行されます。他のすべてのユーザーは、QPADEVxxxx というシステム・デフォルト命名規則を使用し、IBM 提供のサブシステム QINTER で実行されます。

QINTER のカスタマイズ:

サブシステムの独自のセットの使用を開始すると、QINTER を使用する必要はなくなります。ただし、QINTER を継続して使用する理由がある場合は、他のサブシステムの下で実行するワークステーションを割り振らないよう QINTER がセットアップされていることを確認する必要があります。これは、以下の 2 通りの方法で行うことができます。

*ALL ワークステーション項目を QINTER から除去する:

1. *ALL ワークステーション項目を QINTER から除去してから、QINTER が割り振る装置を指定する特定のワークステーション項目を追加します。 *ALL のワークステーション・タイプ項目を除去すると、QINTER がすべてのワークステーションを割り振ることはなくなります。
2. DSP* という名前の装置のワークステーション項目を追加して、すべての平衡型接続ディスプレイ装置を継続して QINTER に割り振れるようにします。

この例では、平衡型接続ディスプレイ装置は継続して QINTER で実行されます。QINTER が他の装置を割り振ることはありません。

```
RMVWSE SBSD(QGPL/QINTER) WRKSTNTYPE(*ALL)
ADDWSE SBSD(QGPL/QINTER) WRKSTN(DSP*)
```

別の方法

ワークステーション項目を追加して、他のサブシステムに割り振られた装置を割り振らないよう QINTER に通知します。ただし、サブシステムに割り振られていない他の装置を QINTER が継続して割り振ることを許可してください。これは、*ALL のワークステーション・タイプ項目を QINTER サブシステムに保持し、さまざまなサブシステムに割り振られる装置の AT パラメーターを使用してワークステーション名前項目を追加します。

```
ADDWSE SBSD(QGPL/QINTER) WRKSTN(PGMR*) AT(*ENTER)
ADDWSE SBSD(QGPL/QINTER) WRKSTN(ORDERENT*) AT(*ENTER)
```

注: システムの装置記述の数が 1 つのサブシステムが処理できる最大数を超過する場合は、この方法を使用できません。

コンソールの構成:

最後の、しかし QINTER に関する非常に重要な考慮事項は、コンソールのワークステーション・タイプ項目 *CONS です。コンソールでの他のユーザーのサインオンを不意に妨げていないことを確認してください。これは、コンソールのどのワークステーション項目もカスタム対話式サブシステムに追加しないことで、発生を防ぐことができます。

システムには、コンソール (*CONS ワークステーション・タイプ項目) のワークステーション項目 AT(*SIGNON) を持つ制御サブシステムが付属しています。 QINTER には、コンソールの AT(*ENTER) ワークステーション・タイプ項目があります。

適切な習慣として、コンソールは必ず制御サブシステムで実行し、コンソール・ジョブを他の対話式サブシステムに転送しないようにします。これにより、コンソールのユーザーはそれぞれのジョブが意図せずに終了することを避けられます。

たとえば、コンソールのユーザーがそのジョブを INTER1 に転送してそのことを忘れてしまい、後からシステム終了 (ENDSYS) コマンドを実行してバックアップ処理の準備を進めた場合、そのコンソール・ジョブも終了してしまいます。 これはほとんどの場合、オペレーターが意図したものではありません。

特定のサブシステムへのユーザーの割り当て:

いくつかの手法を用いて、装置名を割り当てた後、その装置名をユーザーと関連付けることができます。これが完了したら、ワークステーション項目を使用して、ユーザーを正しいサブシステムに割り当てることができます。

システムには、表示セッションに使用されるデフォルトの命名規則があります。これは、複数のサブシステムにわたってユーザー・プロファイル別にワークステーション項目を経路指定する際に不十分な場合があります。

独自の装置命名規則を割り振りおよび管理することによってシステムを変更し、システムのデフォルトの動作を拡張することができます。これは、いくつかの方法でを行うことができます。それぞれの方法に、特有の利点と欠点があります。

関連概念

22 ページの『ワークステーション装置はどのように割り振られるか』
サブシステムは、AT(*SIGNON) ワークステーション項目のサブシステム記述内のすべてのワークステーション装置を割り振ろうとします。

関連情報

経験報告: サブシステム構成

Telnet 出口点プログラムの使用

Telnet 装置初期化および端末出口点:

Telnet 装置初期化および端末出口点。これらの出口点には、システムにサインオンするクライアントを基にした装置名を割り振る機能があります。

出口点は、クライアント IP アドレスおよびユーザー・プロファイル名を (追加情報に加えて) 提供します。次いで、クライアントに使用される装置記述にクライアントを独自にマッピングできます。

装置初期化出口点を使用すると、サインオン・パネルをバイパスすることもできます。

これらの出口点を使用して装置命名規則を管理する方法の利点は、すべてのクライアント用のシステムを中央制御できることです。

欠点は、プログラミング・スキルが必要なことです。

装置選択出口点:

この出口点により、自動的に作成された仮想装置および仮想コントローラーに使用される命名規則、および特殊要求に使用される自動作成限度を指定できます。

この出口点を使用して、Telnet、5250 ディスプレイ装置パススルー、および仮想端末 API に使用される、自動的に作成される装置のさまざまな命名規則を指定できます。

加えて、パススルー装置および Telnet (QAUTOVRT) システム値をさらに厳密な方法で管理できます。たとえば、Telnet の自動的に作成された装置に対してある値を許可し、5250 ディスプレイ装置パススルー装置に対して別の値を許可することができます。

この出口点により、装置に使用されるデフォルトの命名規則 (QPADEV* など) を制御できるようになりますが、これだけでは、特定ユーザーの特定の装置を指定することはできません。 この出口点は、システムへの接続にさまざまな方法 (Telnet、5250 ディスプレイ装置パススルー、WebFacing など) を併用している場合に特に役立ちます。その理由は、これによりさまざまな装置の命名規則と、さまざまなアクセス方式に対する厳密な QAUTOVRT 管理を使用できるからです。

PC5250 (System i Access) ワークステーション ID サポート:

特定のワークステーション名と接続するよう System i Access を構成できます。 このウィンドウで「ヘルプ」ボタンをクリックすると、ワークステーション ID を指定するためのさまざまなオプション (指定した名前が既に使用されている場合に新しい名前を生成するなど) が表示されます。

この方法の欠点は、サーバーに接続するすべてのクライアントで PC5250 構成設定を管理する必要があることです。

OS/400 Telnet Client:

OS/400® Telnet Client コマンド (STRTCPTELN または TELNET) を使用して、サーバー・システムにサインオンするために使用する装置名を指定できます。

デフォルトのアプローチの欠点は、STRTCPTELN (TELNET) コマンドのすべての使用で、リモート仮想ディスプレイの値を適切に指定していることを確認する必要があるということです。この問題を軽減するために、STRTCPTELN コマンドのカスタム・バージョンを作成して、リモート仮想端末ディスプレイの値を確認し、IBM 提供のコマンドを開始することができます。

仮想コントローラーおよび装置の手動作成:

仮想コントローラーおよび装置を手動で作成できます。

Telnet 用の仮想装置の作成についての詳細は、i5/OS Information Center の『Telnet サーバーの構成 (Configure the Telnet Server)』というトピックを参照してください。

コントローラーおよび装置の名前を管理できますが、特定の装置を特定のユーザーにマップすることはできません。

制御サブシステムの作成

IBM 提供の 2 つの完全な制御サブシステム構成として、QBASE (デフォルトの制御サブシステム) および QCTL があります。 システム上では、一時点で 1 つの制御サブシステムだけをアクティブにできます。一般に、IBM 提供のサブシステム構成は、たいていのビジネス・ニーズには十分であるはずです。 ただし、制御サブシステムのユーザー固有のバージョンを作成したり、それを企業の固有のニーズにさらに厳密に合わせて構成することができます。

IBM 提供の制御サブシステム QBASE または QCTL を、ユーザー固有の制御サブシステムを作成するためのモデルとして使用します。

注: ユーザー固有の制御サブシステムを作成する場合は、QBASE または QCTL 以外の名前を使用してください。

制御サブシステムのサブシステム記述には、以下を含める必要があります。

- 以下を含む経路指定項目:
  - 経路指定データとして *ANY または QCMDI のいずれか
  - 呼び出すプログラムとして QSYS/QCMD
  - クラス QSYS/QCTL またはユーザー定義クラス。 (これは、一般的にはシステム・オペレーターであるユーザーが、補助記憶域しきい値に達した場合の記憶域の解放などを実行するためのコマンドを入力できるようにしておく必要があるからです。)
- タイプ *SIGNON のコンソールのワークステーション項目 (*SIGNON は AT パラメーターの値であり、ワークステーション項目追加 (ADDWSE) コマンド上で指定されます。)

  *SIGNON 値は、サブシステムの開始時にワークステーションでサインオン画面が表示されることを示します。 この要件により、サブシステムはシステムおよびサブシステム・レベルのコマンド入力用の対話式装置を必ず持つことになります。 システム終了 (ENDSYS) コマンドは、i5/OS ライセンス・プログラムを終了し、制御サブシステムのコンソールで単一セッション (またはサインオン画面) に移ります。 コンソールのワークステーション項目を含まないサブシステム記述は、制御サブシステムとして開始することはできません。
- もう 1 つのワークステーションの項目:

  これは、制御入力の代替ソースを提供します。コンソール問題が在籍 IPL 時に検出され、コンソール問題発生時 (QSCPFCONS) システム値が 1 に設定されている場合、IPL は不在モードで継続されます。 次いで、制御サブシステムのサブシステム記述にもう 1 つのワークステーションのワークステーション項目が入り、その代替ワークステーションを使用できます。
- 以下を含む経路指定項目:
  - 呼び出すプログラムとして QSYS/QARDRIVE
  - およびクラスとして QSYS/QCTL

制御サブシステムの作成後に、制御サブシステム/ライブラリー (QCTLSBSD) システム値を以下のように変更します (記述の名前は QGPL/QCTLA と想定しています):

```
CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTLA QGPL')
```

変更は次回の IPL 時に有効になります。

関連概念

12 ページの『制御サブシステム』
制御サブシステムは、システムの開始時に自動的に開始される対話式サブシステムであり、システム・オペレーターがシステム・コンソールを使用してシステムの制御に使用するサブシステムです。これは、サブシステム/ライブラリー制御 (QCTLSBSD) システム値で識別されます。

関連情報

経験報告: 制限状態

システムを制限状態にする

制御サブシステムを含むすべてのサブシステムが終了する場合、システムは制限状態になります。対話式ワークステーションから以下の 2 つコマンドのいずれかを使用すると、システムを制限状態にすることができます。

コマンド: *ALL パラメーターを指定したサブシステム終了 (ENDSBS SBS(*ALL))

コマンド: システム終了 (ENDSYS)

重要: ENDSBS または ENDSYS コマンドは、制御サブシステムの対話式ジョブから、および制御サブシステム記述の項目が AT(*SIGNON) を指定しているワークステーションからのみ発行してください。 このコマンドを発行した対話式ジョブは、制御サブシステムが制限状態になっても引き続きアクティブ状態を保ちます。このコマンドを発行したジョブが、(システム要求キーまたは TFRSECJOB コマンドを使用して) ワークステーションでアクティブ状態にある 2 つのジョブのいずれかである場合、いずれのジョブも強制的に終了されることはありません。 制御サブシステムは、こうしたジョブの 1 つを終了するまでは制限状態を終了しません。また抑止されたグループ・ジョブは、(グループ・ジョブが終了するまで) 制御サブシステムが終了しないようにします。

システムが制限状態の場合、システム上のほとんどのアクティビティーは終了しており、1 つのワークステーションだけがアクティブです。システム保管 (SAVSYS) や記憶域の再利用 (RCLSTG) などのコマンドを実行する場合、システムはこの状態でなければなりません。

装置の問題を診断するための一部のプログラムでも、システムが制限状態にあることが必要です。制限状態を終了するには、制御サブシステムを再び開始しなければなりません。

関連概念

12 ページの『制御サブシステム』
制御サブシステムは、システムの開始時に自動的に開始される対話式サブシステムであり、システム・オペレーターがシステム・コンソールを使用してシステムの制御に使用するサブシステムです。これは、サブシステム/ライブラリー制御 (QCTLSBSD) システム値で識別されます。

関連情報

経験報告: 制限状態

メモリー・プールの管理

ジョブが効率的に完了するためメモリーが十分あるようにするのは重要なことです。サブシステム A にメモリーが過剰に与えられて、サブシステム B ではメモリーが不足する場合、サブシステム B のジョブは効率よく実行されない可能性があります。以下の情報では、メモリー・プールの管理に関連した種々のタスクについて説明されています。

関連概念

25 ページの『メモリー・プール』
メモリー・プールとは、メイン・メモリーまたは記憶域の論理的な区画のことで、ジョブまたはジョブ・グループの処理のために予約されています。システム上のすべての主記憶域はメモリー・プールと呼ばれる論理的な割り振りに区分されます。デフォルトでは、システムはメモリー・プールへのデータおよびプログラムの転送も管理します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

メモリー・プール情報の表示

System i ナビゲーター または文字ベース・インターフェースを使用して、システム上にあるメモリー・プールについての情報を表示できます。

関連概念

29 ページの『メモリー・プールの割り振り』
サブシステムを開始すると、システムは、開始されたサブシステムのサブシステム記述で定義されたユーザー定義の記憶域プールを割り振ろうとします。

30 ページの『メモリー・プールのアクティビティー・レベル』
メモリー・プールのアクティビティー・レベルは、メモリー・プール内で CPU を同時に使用できるスレッドの数です。こうすることによって、システム・リソースを効果的に使用できます。アクティビティー・レベルの制御はシステムが管理します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:
System i ナビゲーターで、**「ユーザー接続」** → **「接続 (connection)」** → **「実行管理機能」** → **「メモリー・プール」** → **「アクティブ・プール」**または**「共用プール」**の順に展開します。

アクティブ・プール・コンテナーは、共用プールおよび専用プールがアクティブであればその両方を表示します。共用プール・コンテナーは、その現在の状態に関係なく、すべての共用プールを表示します。非アクティブな専用プールは、サブシステムによってアクティブにされるまでは、プール定義の範囲を越えて存在することはありません。したがって、それらは System i ナビゲーター を使用して表示することはできません。

文字ベース・インターフェース:

コマンド: サブシステム記述表示 (DSPSBSD)

オプション 2 -「プール定義 (Pool Definitions)」を使用して、このサブシステム定義内に存在するすべての専用および共用プール定義を表示します。

コマンド: 共用プール処理 (WRKSHRPOOL)

メモリー・プールを使用したサブシステム数の判別

複数のジョブを実行するために、サブシステムには特定の比率でメモリーが割り振られます。いくつの異なるサブシステムが同じメモリー・プールからプルしているかを知っていることは重要です。いくつのサブシステムがプールにジョブを送り出しているか、またいくつのジョブがプールで実行されているかを知ると、プールのサイズとアクティビティー・レベルを調整して、リソースの競合を減らしたいと場合もあります。

関連概念

29 ページの『メモリー・プールの割り振り』
サブシステムを開始すると、システムは、開始されたサブシステムのサブシステム記述で定義されたユーザー定義の記憶域プールを割り振ろうとします。

30 ページの『メモリー・プールのアクティビティー・レベル』
メモリー・プールのアクティビティー・レベルは、メモリー・プール内で CPU を同時に使用できるスレッドの数です。こうすることによって、システム・リソースを効果的に使用できます。アクティビティー・レベルの制御はシステムが管理します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:
System i ナビゲーター を使用して、メモリー・プールを使用しているサブシステムの数をモニターするには、以下のようにします。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (Connection)**」 → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 処理するメモリー・プールを右マウス・ボタン・クリックして、「**サブシステム**」をクリックします。

   このウィンドウから、同一のメモリーを使用してジョブを実行しているサブシステムの数を判断することができます。

文字ベース・インターフェース:
コマンド: サブシステム処理 (WRKSBS)

このコマンドは、すべてのサブシステムおよびその対応するプールのリストを表示します。

メモリー・プール内のジョブ数の判別

System i ナビゲーター には、メモリー・プール内で現在実行中のジョブのリストを素早く表示する手段が備えられています。

メモリー・プール内のジョブの数を判別するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 使用するメモリー・プールを右マウス・ボタン・クリックし、「**ジョブ**」をクリックします。メモリー・プールにあるジョブのリストを示したウィンドウが表示されます。

   「スレッド・カウント (Thread Count)」の欄を表示すると、メモリー・プール内のスレッドの数も分かります。スレッド・カウントから、メモリー・プール内のアクティビティーの量についてさらに知ることができます。

   この時点から、アクティブ・ジョブ区域またはサーバー・ジョブ区域にいるときと同じ機能を実行することができます。

関連概念

29 ページの『メモリー・プールの割り振り』
サブシステムを開始すると、システムは、開始されたサブシステムのサブシステム記述で定義されたユーザー定義の記憶域プールを割り振ろうとします。

30 ページの『メモリー・プールのアクティビティー・レベル』
メモリー・プールのアクティビティー・レベルは、メモリー・プール内で CPU を同時に使用できるスレッドの数です。こうすることによって、システム・リソースを効果的に使用できます。アクティビティー・レベルの制御はシステムが管理します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

単一ジョブを実行しているプールの判別

期待通りに実行できないジョブがある場合、そのジョブを実行中のメモリー・プールを確認することもできます。単一ジョブが実行されているプールを判別するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

ジョブを実行しているプールを識別したなら、メモリー・プール情報を表示して、変更を加える必要があるかどうかを判断できます。たとえば、生じるページングが多すぎる場合、メモリー・プールを大きくする必要があるかもしれません。パフォーマンスが低い別の可能性としては、プール内に他のジョブが多すぎる場合があり、このジョブを別のプールに経路指定する必要があります。

関連概念

29 ページの『メモリー・プールの割り振り』
サブシステムを開始すると、システムは、開始されたサブシステムのサブシステム記述で定義されたユーザー定義の記憶域プールを割り振ろうとします。

30 ページの『メモリー・プールのアクティビティー・レベル』
メモリー・プールのアクティビティー・レベルは、メモリー・プール内で CPU を同時に使用できるスレッドの数です。こうすることによって、システム・リソースを効果的に使用できます。アクティビティー・レベルの制御はシステムが管理します。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:

System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーターで、処理するジョブのタイプに応じて、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」または「**サーバー・ジョブ**」の順に展開します。
2. 表示するメモリー・プールのジョブを見つけます。
3. 「**ジョブ名**」を右マウス・ボタン・クリックして、「**プロパティー**」をクリックします。
4. 「**Resources (リソース)**」タブをクリックします。「Job Properties - Resources (ジョブ・プロパティー - リソース)」ウィンドウに、そのジョブのメモリー・プールに関する特定の情報が表示されます。

文字ベース・インターフェース:

コマンド: ジョブ処理 (WRKJOB)

オプション 1: Display Job Status Attributes (ジョブ状況属性の表示)

「Subsystem pool ID (サブシステム・プール ID)」フィールドには、ジョブが実行されているサブシステム用に定義されたプールの名前が含まれます。このフィールドは、表示が要求された時点でアクティブ状態にないジョブに関してはブランクになります。また、システム・ジョブ (タイプ SYS)、サブシステム内で実行されていないサブシステム・モニター・ジョブ (タイプ SBS)、基本メモリー・プールで実行されているバッチ即時ジョブ (BCI) に関してもブランクになります。

コマンド: アクティブ・ジョブ処理 (WRKACTJOB)

WRKACTJOB コマンドを使用すると、アクティブ・ジョブのシステム・プール ID を表示できます。

共用プールのチューニング・パラメーターの管理

共用プールのチューニング・パラメーターを管理するには、System i ナビゲーター または文字ベース・インターフェースのコマンドを使用します。

関連概念

27 ページの『プール番号付けのスキーム』
プールには 2 つの番号付けスキームがあり、1 つはサブシステム内で使用し、もう 1 つはシステム全体で使用します。サブシステムは、使用するプールを参照する一連の番号を用います。ですから、サブシステム記述を作成または変更する場合、1 つ以上のプールを定義し、それらに 1、2、3 のようにラベル付けを行えます。これはサブシステム・プールの指定であって、システム状況処理 (WRKSYSSTS) で表示されるプール番号には対応していません。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:

System i ナビゲーターを使用してチューニング・パラメーターにアクセスするには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 調整するプールを右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。
3. 「**Tuning (チューニング)**」タブをクリックします。

「Shared Properties - Tuning (共用プロパティー - チューニング)」ウィンドウで、プール割り振り比率、毎秒ごとのページ不在、および優先順位などの特定の値を手動で調整できます。

文字ベース・インターフェース:

コマンド: 共用記憶域プールの処理 (**WRKSHRPOOL**)

「**Option 11 - Display tuning data (オプション 11 - チューニング・データの表示)**」を選択します。

プールの構成の管理

プール・サイズ、アクティビティー・レベル、またはページング・オプションを変更するには、 System i ナビゲーター または文字ベース・インターフェースのコマンドを使用します。

関連概念

27 ページの『プール番号付けのスキーム』
プールには 2 つの番号付けスキームがあり、1 つはサブシステム内で使用し、もう 1 つはシステム全体で使用します。サブシステムは、使用するプールを参照する一連の番号を用います。ですから、サブシステム記述を作成または変更する場合、1 つ以上のプールを定義し、それらに 1、2、3 のようにラベル付けを行えます。これはサブシステム・プールの指定であって、システム状況処理 (WRKSYSSTS) で表示されるプール番号には対応していません。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:

System i ナビゲーターを使用して共用プールの構成値にアクセスするには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**メモリー・プール**」 → 「**アクティブ・プール**」または「**共用プール**」の順に展開します。
2. 調整するプールを右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。
3. 「**構成**」タブをクリックします。

「Shared Properties - Configuration (共用プロパティー - 構成)」ウィンドウで、プール・サイズ、アクティビティー・レベル、またはページング・オプションなどの特定の値を手動で調整できます。

文字ベース・インターフェース:

コマンド: 共用記憶域プールの処理 (**WRKSHRPOOL**)

メモリー・プール・サイズの変更

メモリー・プールのサイズ変更は、サブシステムが処理できる作業の量に直接影響します。多くのメモリーが与えられれば、サブシステムはさらに多くの作業を行えるようになります。メモリー・プールのパラメーターの変更を開始する前には、システムを注意深くモニターすることが大切です。さらに、再調整が必要になる場合もあるため、調整後もこれらのレベルを定期的に確認してください。

手動でのメモリー・プールのサイズ変更を開始する場合、 その前に必ずシステムの調整機能をオフにしてください。システムの調整機能は、共用メモリー・プールのサイズを、システムが実行している作業の量に自動的に調整してしまうためです。システムの調整機能をオフにしていないと、手動で行った変更は調整機能によってさらに自動的に変更されてしまいます。

システムの調整機能をオフにするには、メモリー・プールおよびアクティビティー・レベルの自動調整 (QPFRADJ) システム値を 0 (0 = 調整なし) に変更します。

関連概念

27 ページの『プール番号付けのスキーム』

プールには 2 つの番号付けスキームがあり、1 つはサブシステム内で使用し、もう 1 つはシステム全体で使用します。サブシステムは、使用するプールを参照する一連の番号を用います。ですから、サブシステム記述を作成または変更する場合、1 つ以上のプールを定義し、それらに 1、2、3 のようにラベル付けを行えます。これはサブシステム・プールの指定であって、システム状況処理 (WRKSYSSTS) で表示されるプール番号には対応していません。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

System i ナビゲーター:

1. System i ナビゲーターで、**「ユーザー接続」** → **「接続 (Connection)」** → **「実行管理機能」** → **「メモリー・プール」** → **「アクティブ・プール」または「共用プール」**の順に展開します。
2. 処理するメモリー・プール (たとえば「対話式」) を右マウス・ボタン・クリックして、**「プロパティー」**を右マウス・ボタン・クリックします。**「メモリー・プールのプロパティー (Memory Pool Properties)」**ウィンドウが表示されます。
3. **「プロパティー」**ウィンドウの「構成」タブで、定義済みのメモリー・サイズを変更できます。定義済みのメモリーは、プールが使用できるメモリーの最大サイズです。ここに指定される数は、そのプールが機能を果たすサブシステムをサポートするのに必要なメモリー・サイズを反映しているべきです。

注: 基本プールは、メモリー・サイズが定義されていない唯一のメモリー・プールです。代わりとして、実行のために必要なメモリーの最小サイズが指定されています。基本プールには、他に割り振られないすべてのメモリーが含まれます。たとえば、システムのメモリーの総量が 1000 MB とします。そのうちの 250 MB がマシン・プールに割り振られ、さらに 250 MB が対話式プールに割り振られているとします。すると、500 MB はどこにも割り振られていません。この未割り振りのメモリーは、必要が生じるまで基本プールに保管されています。

メモリーを移動させるときには注意が必要です。あるプールから別のプールへメモリーを移動する場合、1 つのサブシステムを修正するだけですが、他のサブシステムに問題を引き起こすことがあり、システム・パフォーマンスを低下させることもあります。

文字ベース・インターフェース:
コマンド: システム値変更 (CHGSYSVAL)

例: 以下は、マシン・プールのサイズを変更します。

```
CHGSYSVAL QMCHPOOL 'new-size-in-KB'
```

これは、WRKSYSTS 画面のプール 1 に対応します。

例: 以下は、基本プールの最小サイズを変更します。

```
CHGSYSVAL QBASPOOL 'new-minimum-size-in-KB'
```

これは、WRKSYSSTS 画面のプール 2 に対応します。

注: QBASPOOL システム値は、基本プールの最小サイズを制御するだけです。基本プールには、他のプールに割り振られていないすべての記憶域が含まれます。

共用プールのサイズの変更:

コマンド: 共用記憶域プール変更 (CHGSHRPOOL)

共用プールの変更は、共用プールがアクティブであり、十分な記憶域が使用できる場合は、即時に有効になります。

コマンド: 共用記憶域プール処理 (WRKSHRPOOL)

このコマンドにより、共用プールの名前および状況情報にアクセスできます。メニュー・オプションを使用することにより、プール・サイズおよび最大アクティビティー・レベルの値を変更できます。

専用メモリー・プールの作成

専用メモリー・プール (ユーザー定義メモリー・プールとしても知られる) は、IBM 提供のサブシステムまたはユーザー定義サブシステムで使用できます。サブシステムは最大で 10 のメモリー・プール定義を持つことができます。専用メモリー・プールはサブシステム記述内に作成します。

専用メモリー・プールを作成するには、文字ベース・インターフェースを使用します。

コマンド: サブシステム記述作成 (CRTSBSD) POOLS パラメーター。

コマンド: サブシステム記述変更 (CHGSBSD) POOLS パラメーター。

注: それぞれのサブシステム記述は最大で 10 のユーザー定義メモリー・プールを持つことができますが、いつでもアクティブにできるのは 64 メモリー・プール以内という操作制限があります。 (これには基本メモリー・プールおよびマシン・メモリー・プールが含まれます。) サブシステムのすべてのメモリー・プールが割り振られる前に最大割り振り制限に達した場合、メモリー・プールを引き続き必要とする経路指定ステップには基本プールが使用されます。

関連概念

26 ページの『メモリー・プールの種類』
システム上のすべての主記憶域はメモリー・プール と呼ばれる論理的な割り振りに区分されます。システムのすべてのメモリー・プールは、私用または共用のいずれかです。専用メモリー・プール、共用メモリー・プール、および特殊共用メモリー・プールがあります。私用プールと共用プールを合わせて最大 64 個のメモリー・プールまで、同時にアクティブ状態にすることができます。

関連情報

システム・パフォーマンスの管理

パフォーマンスのチューニング

パフォーマンス管理用のアプリケーション

経験報告: パフォーマンス・アジャスター (QPFRADJ)

パフォーマンス・システム値: マシン記憶域プール・サイズ

パフォーマンス・システム値: 基本記憶域プールの最小サイズ

パフォーマンス・システム値: 基本記憶域プール有資格スレッドの最大数

ジョブ待ち行列の管理

システムでの作業を管理する際、ジョブ待ち行列で待機中のジョブを扱う必要があることに気付く場合があります。ジョブをすぐに実行することが必要な場合もあれば、ジョブを優先順位の低い状態で待ち行列に入れることもあります。あるいは、サブシステム上で保守を実行し、特定のサブシステムに関連付けられていない待ち行列にすべてのジョブを移動する必要がある場合もあります。

以下の情報では、こうしたタイプの管理タスクを実行する方法について説明されています。

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

サブシステムへのジョブ待ち行列の割り当て

ジョブ待ち行列項目をサブシステム記述に割り当てるには、文字ベース・インターフェースを使用します。

コマンド: ジョブ待ち行列項目追加 (ADDJOBQE)

このコマンドのパラメーターは、以下を指定します。

- このジョブ待ち行列上で同時にアクティブにできるジョブの数 (MAXACT)
- このジョブ待ち行列の作業がサブシステムによって処理される順序 (SEQNBR)
- 9 つのレベルの各優先順位に対して一度にアクティブにできるジョブの数 (MAXPTYn) (n は 1 から 9)

例: 以下の例では、TEST サブシステム記述内の JOBQA ジョブ待ち行列のジョブ待ち行列項目を追加します。このジョブ待ち行列上でアクティブにできるジョブの最大数はなく、作業は順序番号 5 で処理されます。

```
ADDJOBQE SBSD(TEST) JOBQ(LIBA/JOBQA) MAXACT(*NOMAX) SEQNBR(5)
```

関連概念

70 ページの『ジョブ待ち行列はどのように機能するか』
ジョブ待ち行列は、ジョブ待ち行列項目によってサブシステムで割り振られます。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステムはその開始時に、待ち行列上のジョブを処理します。

関連タスク

192 ページの『ジョブ待ち行列項目の除去』
文字ベース・インターフェースを使用して、ジョブ待ち行列項目をサブシステム記述から除去できます。ジョブ待ち行列項目がサブシステム記述から除去される時に、そのジョブ待ち行列上のジョブは待ち行列に残っています。現在アクティブ状態のいずれかのジョブがジョブ待ち行列から開始された場合には、ジョブ待ち行列項目を除去することはできません。

サブシステムは複数のジョブ待ち行列をどのように処理するか:

サブシステムが複数のジョブ待ち行列をどのように処理するかを明らかにするために、以下のシナリオを考慮してください。

ジョブ待ち行列 A (SEQNBR=10)

ジョブ 1

ジョブ 2

ジョブ 3

ジョブ待ち行列 B (SEQNBR=20)

ジョブ 4

ジョブ 5

ジョブ 6

ジョブ待ち行列 C (SEQNBR=30)

ジョブ 7

ジョブ 8

ジョブ 9

このシナリオのそれぞれのジョブ待ち行列項目は、MAXACT(*NOMAX) と指定されます。ジョブ待ち行列項目が最小の順序番号であるため、サブシステムはまずジョブ待ち行列 **A** からジョブを選択します。サブ

システム内のジョブの最大数が 3 の場合 (サブシステム記述作成 (CRTSBSD) コマンドの MAXJOBS(3) パラメーター)、ジョブ待ち行列 **A** から、同時にアクティブにするすべてのジョブを選択できます。

3 つのジョブのいずれかが完了すると、アクティビティー・レベルは最大ではなくなります。したがって、次に小さい順序番号を持つのがジョブ待ち行列 **B** であるので、それから新規ジョブが選択されます (新規ジョブがジョブ待ち行列 **A** に追加されていないと想定しています)。それぞれのジョブ待ち行列項目は MAXACT(*NOMAX) を指定しているので、MAXACT 値によりジョブが開始できなくなることはありません。それぞれのジョブ待ち行列項目に MAXACT(1) が指定されていれば、ジョブ 1、4、および 7 が開始されます。ジョブ待ち行列項目 **A** が MAXACT(2) と指定されていれば、ジョブ 1、2、および 4 が開始されます。

関連概念

70 ページの『ジョブ待ち行列はどのように機能するか』
ジョブ待ち行列は、ジョブ待ち行列項目によってサブシステムで割り振られます。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステムはその開始時に、待ち行列上のジョブを処理します。

ジョブ待ち行列内で同時に実行するジョブ数の変更

QBASE サブシステムには、QBATCH ジョブ待ち行列用のジョブ待ち行列項目が付属します。この項目では、一度に 1 つのバッチ・ジョブしか実行できません。そのジョブ待ち行列から複数のバッチ・ジョブを同時に実行したい場合は、ジョブ待ち行列項目を変更する必要があります。

ジョブ待ち行列から同時に実行するジョブ数を変更するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ待ち行列項目変更 (CHGJOBQE)

例: 以下のコマンドにより、QBATCH ジョブ待ち行列からの 2 つのバッチ・ジョブを、QBASE サブシステム内で同時に実行できます。 (このコマンドはいつでも発行可能であり、即時に有効になります。)

```
CHGJOBQE SBSD(QBASE) JOBQ(QBATCH) MAXACT(2)
```

関連概念

73 ページの『ジョブを複数のジョブ待ち行列から取り出す方法』
サブシステムは、ジョブ待ち行列からのジョブを順序番号に基づいて処理します。サブシステムは複数のジョブ待ち行列項目を持つことができ、そのため複数のジョブ待ち行列を割り振ることができます。

70 ページの『ジョブ待ち行列からジョブが取り出される方法』
ジョブ待ち行列からジョブが選択され、開始される方法を決めるさまざまな要素があります。

71 ページの『ジョブ待ち行列項目』
ジョブ待ち行列項目は、サブシステムで実行するジョブの選択元になるジョブ待ち行列を示します。ジョブ待ち行列項目には、ジョブ待ち行列の処理方法を制御する 5 つのパラメーターがあります。

ジョブ待ち行列の消去

ジョブ待ち行列を消去すると、待ち行列上のすべてのジョブが削除されます。これには、保留状態のすべてのジョブが含まれます。ジョブ待ち行列を消去するには、System i ナビゲーター ナビゲーターまたは文字ベース・インターフェースを使用できます。 実行中のジョブは、アクティブ・ジョブと見なされ、待ち行列上にはないので、影響を受けません。

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョ

ブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

System i ナビゲーター:
ジョブ待ち行列を消去するために System i ナビゲーター ナビゲーターを使用するには、以下のステップに従います。

1. **「ユーザー接続」 → 「接続 (connection)」 → 「ジョブ待ち行列」 → 「アクティブ・ジョブ待ち行列」または「すべてのジョブ待ち行列」**の順に展開します。
2. ジョブ待ち行列を右マウス・ボタン・クリックして、**「消去」**をクリックします。 「消去の確認 (Confirm Clear)」ウィンドウが表示され、待ち行列の消去時にジョブ・ログを生成するかどうかを指定できます。

文字ベース・インターフェース:
コマンド: ジョブ待ち行列消去 (CLRJOBQ)

例: このコマンドは、IBM 提供のジョブ待ち行列 QBATCH に現在あるすべてのジョブを除去します。現在読み取り中のジョブは影響を受けません。

```
CLRJOBQ JOBQ(QGPL/QBATCH)
```

ジョブ待ち行列の作成

ジョブ待ち行列を作成するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ待ち行列作成 (CRTJOBQ)

例: 以下の例では、LIBA ライブラリー内に JOBQA と呼ばれるジョブ待ち行列を作成します。

```
CRTJOBQ JOBQ(LIBA/JOBQA) TEXT('test job queue')
```

ジョブ待ち行列の作成後には、ジョブの実行前にそれをサブシステムに割り当てる必要があります。ジョブ待ち行列をサブシステムに割り当てるには、ジョブ待ち行列項目をサブシステム記述に追加します。

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

70 ページの『ジョブ待ち行列はどのように機能するか』
ジョブ待ち行列は、ジョブ待ち行列項目によってサブシステムで割り振られます。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステムはその開始時に、待ち行列上のジョブを処理します。

ジョブ待ち行列の削除

ジョブ待ち行列を削除するには、文字ベース・インターフェースを使用します。

制限事項:

- 削除されるジョブ待ち行列には、項目を含めることはできません。待ち行列上のすべてのジョブは、完了、削除、または別のジョブ待ち行列への移動が必要です。
- ジョブ待ち行列に対してサブシステムをアクティブにすることはできません。

ジョブ待ち行列を削除するには複数の方法があります。ここでは 2 種類の方法をリストしていますが、ジョブ数および状況を表示するので、WRKJOBQ コマンドを推奨します。

コマンド: ジョブ待ち行列処理 (WRKJOBQ)

ジョブの数が 0 の場合、オプション 4=「削除」を使用して、ジョブ待ち行列をライブラリーから削除できます。

DLTJOBQ を自動スクリプトと共に使用して、環境をクリーンアップします。このコマンドのデフォルトの動作は、ライブラリー・リストを検索し、指定した名前と一致する最初のジョブ待ち行列を削除することなので、この方法を使用する場合は注意してください。異なるライブラリー内に同じ名前の 2 つのジョブ待ち行列がある場合、正しくない方のジョブ待ち行列を削除することになるかもしれません。この動作は、特定のライブラリーを指定することで指定変更できます。

コマンド: ジョブ待ち行列削除 (DLTJOBQ)

例: このコマンドは、ライブラリー SPECIALLIB 内のジョブ待ち行列 SPECIALJQ を削除します。

```
DLTJOBQ JOBQ(SPECIALLIB/SPECIALJQ)
```

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

ジョブ待ち行列が割り振られているサブシステムの判別

どのサブシステムがジョブ待ち行列を割り振ったかを判別するには、System i ナビゲーター インターフェースまたは文字ベース・インターフェースを使用します。 サブシステムがアクティブになっているジョブ待ち行列は削除できないため、これはジョブ行列を削除する必要がある場合に役立ちます。

関連概念

70 ページの『ジョブ待ち行列はどのように機能するか』
ジョブ待ち行列は、ジョブ待ち行列項目によってサブシステムで割り振られます。サブシステムが開始されていない場合でも、ジョブをジョブ待ち行列に入れることができます。サブシステムはその開始時に、待ち行列上のジョブを処理します。

System i ナビゲーター:

ジョブ待ち行列を割り振ったサブシステムを調べるには、以下の手順に従います。

1. System i ナビゲーターで、**「実行管理機能」** → **「ジョブ待ち行列」** → **「すべてのジョブ待ち行列」**の順に展開します。
2. System i ナビゲーター・インターフェースの右側のペインでジョブ待ち行列を見付けます。 ジョブ待ち行列を割り振ったサブシステムが「サブシステム」列に表示されます。

   (「サブシステム」列が表示されていない場合は、画面に追加します。**「すべての待ち行列」を右クリック** → **「このビューのカスタマイズ」** → **「列」**。)
3. または、ジョブ待ち行列を右クリックして、**「プロパティー」**をクリックします。サブシステムは、「ジョブ待ち行列のプロパティー (Job Queue Properties)」ウィンドウの「一般」ページにリストされています。

文字ベース・インターフェース:

コマンド: WRKJOBQ JOBQ(LIBA/JOBQA) (JOBQA はジョブ待ち行列の名前)

1. コマンド WRKJOBQ JOBQ(LIBA/JOBQA) を入力する。 「ジョブ待ち行列の処理」画面が表示されます。ジョブ待ち行列がシステムに割り振られると、サブシステム記述ファンクション・キーが画面のファンクション・キー域に表示されます。
2. サブシステム記述ファンクション・キーを押す。 「サブシステム記述の処理 (Work with Subsystem Descriptions)」画面が表示され、ジョブ待ち行列が割り振られているサブシステムが表示されます。

ジョブ待ち行列の保留

ジョブ待ち行列を保留状態にすると、現在ジョブ待ち行列で待機中のすべてのジョブの処理が妨げられます。ジョブ待ち行列を保留状態にしても、実行中のジョブには影響を与えません。待ち行列が保留中に追加のジョブをジョブ待ち行列に入れることができますが，それらは処理されません。

ジョブ待ち行列を保留状態にするには、System i ナビゲーター または文字ベース・インターフェースを使用できます。

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

System i ナビゲーター:
System i ナビゲーターで、**「ユーザー接続」 → 「接続 (connection)」 → 「実行管理機能」 → 「ジョブ待ち行列」 → 「アクティブ・ジョブ待ち行列」 → 待ち行列を右クリックする → 「保留」**の順に展開します。

文字ベース・インターフェース:
コマンド: ジョブ待ち行列保留 (HLDJOBQ)

次の例では、ジョブ待ち行列 QBATCH が保留状態にされます。このコマンドの発行時に実行されていないすべてのジョブは、待ち行列が解放されるか消去されるまで保留されます。

```
HLDJOBQ   JOBQ(QBATCH)
```

ジョブ待ち行列の解放

ジョブ待ち行列を解放すると、ジョブ待ち行列が保留された結果として保留されたすべてのジョブも解放されます。ジョブ待ち行列が保留される前に個別のジョブが保留されていた場合は、そのジョブは解放されません。

ジョブ待ち行列を解放するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連概念

68 ページの『ジョブ待ち行列』
ジョブ待ち行列には、サブシステムによる処理を待っているジョブの番号付きリストがあります。ジョブ待ち行列は、バッチ・ジョブがサブシステムでアクティブ状態になる前に、最初に送られる場所です。ジョブは、いくつかの要因が一致するまでそこで保持されます。

System i ナビゲーター:
System i ナビゲーターで、**「ユーザー接続」 → 「接続 (Connection)」 → 「実行管理機能」 → 「ジョブ待ち行列」 → 「すべてのジョブ待ち行列」 → 待ち行列を右クリックする → 「解放」**の順に展開します。

文字ベース・インターフェース:

コマンド: ジョブ待ち行列解放 (RLSJOBQ)

この例は、ジョブ待ち行列 QBATCH を解放します。

```
RLSJOBQ   JOBQ(QBATCH)
```

別のジョブ待ち行列へのジョブの移動

さまざまな理由により、ジョブを別の待ち行列に移動したい場合があります。たとえば、ジョブが長く実行されているため、ジョブが待ち行列内でバックログになることがあります。おそらく、ジョブのスケジュールされた実行時間が高位の優先順位を持つ新しいジョブと競合しているのかもしれません。このような状況を管理する 1 つの方法は、待機中のジョブを使用率のそれほど高くない別の待ち行列に移動することです。

ジョブを待ち行列間で移動するには、System i ナビゲーター インターフェースまたは文字ベース・インターフェースのどちらも使用できます。

関連概念

73 ページの『ジョブを複数のジョブ待ち行列から取り出す方法』
サブシステムは、ジョブ待ち行列からのジョブを順序番号に基づいて処理します。サブシステムは複数のジョブ待ち行列項目を持つことができ、そのため複数のジョブ待ち行列を割り振ることができます。

70 ページの『ジョブ待ち行列からジョブが取り出される方法』
ジョブ待ち行列からジョブが選択され、開始される方法を決めるさまざまな要素があります。

System i ナビゲーター:

System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーターで、「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**すべてのジョブ待ち行列**」の順に展開します。
2. ジョブが現在含まれている待ち行列を見つけて、オープンします。
3. 移動するジョブを右マウス・ボタンでクリックします。 「移動」ウィンドウがオープンし、ターゲット待ち行列を指定できます。

   注: この待ち行列から複数のジョブを移動したい場合、各ジョブをクリックする際 CTRL キーを押したままにします。その後、右マウス・ボタン・クリックして 「**移動**」をクリックします。

- 実行を待機中のジョブは、ターゲット待ち行列上でも同じ相対位置に移動します (たとえば、ジョブ待ち行列優先順位 3 のジョブは、ターゲット待ち行列上で実行を待機中の優先順位 3 の他のジョブの後ろに移動します)。
- 保留されているジョブは、ターゲット待ち行列上でも引き続き保留され、同じ相対位置に置かれます (たとえば、ジョブ待ち行列優先順位 3 の保留ジョブは、ターゲット待ち行列上の優先順位 3 の他の保留ジョブの後ろに移動します)。
- 実行がスケジュールされているジョブは、ターゲット待ち行列に移動される際にそのスケジュールされた時間は未変更のままです。

文字ベース・インターフェース:

コマンド: ジョブ変更 (CHGJOB)

例: 次の例では、ジョブ JOBA がジョブ待ち行列 JOBQB に移動します。

```
CHGJOB JOB(JOBA) JOBQ(LIBA/JOBQB)
```

ジョブのジョブ待ち行列への配置

既存のジョブをある待ち行列から別の待ち行列に移動するか、新規ジョブを投入することによって、ジョブはジョブ待ち行列に入れられます。待ち行列間でジョブを移動するには、System i ナビゲーター を使用します。 新規ジョブを投入するには、文字ベース・インターフェースを使用します。

関連概念

73 ページの『ジョブを複数のジョブ待ち行列から取り出す方法』
サブシステムは、ジョブ待ち行列からのジョブを順序番号に基づいて処理します。サブシステムは複数のジョブ待ち行列項目を持つことができ、そのため複数のジョブ待ち行列を割り振ることができます。

70 ページの『ジョブ待ち行列からジョブが取り出される方法』
ジョブ待ち行列からジョブが選択され、開始される方法を決めるさまざまな要素があります。

System i ナビゲーター:

System i ナビゲーター インターフェースを使用するには、ジョブが別のジョブ待ち行列に存在している必要があります。 この場合、ジョブをある待ち行列から別の待ち行列に移動できます。(新規ジョブをジョブ待ち行列に入れるには、コマンド行インターフェースを使用します。)

1. System i ナビゲーターで、「**実行管理機能**」 → 「**ジョブ待ち行列**」 → 「**すべてのジョブ待ち行列**」の順に展開します。
2. 移動するジョブを右マウス・ボタンでクリックします。 「Move (移動)」ウィンドウが開き、宛先待ち行列を指定できます。

文字ベース・インターフェース:

以下に、新規ジョブを新規ジョブ待ち行列に入れるための文字ベース・インターフェース・メソッドのリストを示します。

- ジョブ投入 (SBMJOB): 実行中のジョブが別のジョブをジョブ待ち行列に投入して、後からバッチ・ジョブとして実行できるようにすることができます。 要求データの 1 つの要素だけを新しいジョブのメッセージ待ち行列に入れることができます。ジョブに使用する経路指定項目が CL コマンド処理プログラム (たとえば、IBM 提供の QCMD プログラム) を指定している場合には、要求データは CL コマンドであってもかまいません。
- ジョブ・スケジュール項目追加 (ADDJOBSCDE): システムは、ジョブ・スケジュール項目で指定された日付および時刻にジョブをジョブ待ち行列に投入します。
- データベース・ジョブ投入 (SBMDBJOB): ジョブをバッチ・ジョブとして実行できるようにジョブ待ち行列に投入します。 入力ストリームは、物理データベース・ファイルから、あるいは単一レコード様式を持つ論理データベース・ファイルから読み取られます。このコマンドでは、このデータベース・ファイルとそのメンバーの名前、使用されるジョブ待ち行列の名前、および投入されるジョブが投入ジョブ処理 (WRKSBMJOB) コマンドによって表示できるかどうかを指定することができます。
- データベース読取プログラム開始 (STRDBRDR): データベースからバッチ入力ストリームを読み取り、1 つ以上のジョブをジョブ待ち行列に入れます。
- ジョブ転送 (TFRJOB): 現行ジョブをアクティブ・サブシステムの別のジョブ待ち行列に移動します。
- バッチ・ジョブ転送 (TFRBCHJOB): 現行ジョブを別のジョブ待ち行列に移動します。

特定のジョブのすべてのジョブ待ち行列の検索

System i ナビゲーター または文字ベース・インターフェースのいずれかを使用して、特定のジョブのジョブ待ち行列を検索できます。

System i ナビゲーター:

1. System i ナビゲーターで、**「ユーザー接続」 → 「接続 (Connection)」 → 「基本操作」 → ジョブを右クリック → 「ビューのカスタマイズ (Customize this View)」 → 「組み込み」**の順に展開します。
2. 「ジョブ - 組み込み (Jobs-Include)」ウィンドウを使用して、表示されるジョブの数を絞り込みます。**「ジョブ待ち行列」**フィールドが「すべて (All)」に設定されていることを確認します。
3. **「OK」**をクリックすると、基準を満たすすべてのジョブが表示されます。

文字ベース・インターフェース:
コマンド: ジョブ待ち行列処理 (WRKJOBQ)

例: 以下の例では、JOBQA ジョブ待ち行列上のすべてのジョブのリストを作成します。

```
WRKJOBQ JOBQ(LIBA/JOBQA)
```

ジョブ待ち行列の名前が分からない場合のジョブの検索:
ジョブ待ち行列の名前が分からない場合は、以下の指示に従います。

1. JOBQ パラメーターを指定せずにコマンドを入力します。 「すべてのジョブ待ち行列の処理 (Work with All Job Queues)」ウィンドウが表示され、許可が与えられているすべてのジョブ待ち行列のリストが示されます。
2. 見つけようとしているジョブが含まれている可能性があるジョブ待ち行列の名前が表示されるまで、このリストをスキャンします。

ジョブ待ち行列内でジョブを見つけた後に、表示したいジョブに対して処理オプションを入力することで、そのジョブを表示できます。「ジョブ処理」画面が表示されます。この画面は、選択したジョブについて入手できるすべての情報を表示するためのいくつかのオプションを備えています。

探しているジョブが分かっている場合、以下のコマンドを入力すると、直接そのジョブの画面に進むことができます。

```
WRKJOB JOB(number/user/name) OPTION(*DFNA)
```

探しているジョブがはっきりとは分からない場合、投入済みジョブ処理 (WRKSBMJOB) またはユーザー・ジョブ処理 (WRKUSRJOB) が役立つ場合があります。

ジョブ待ち行列の優先順位の指定

ジョブ待ち行列がサブシステムによって処理される順序を指定するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ待ち行列項目追加 (ADDJOBQE)

このコマンドのパラメーターは、以下を指定します。

- このジョブ待ち行列上で同時にアクティブ状態にできるジョブの最大数 (MAXACT)
- このジョブ待ち行列の作業がサブシステムによって処理される順序 (SEQNBR)
- 優先順位の 9 つのレベルごとに同時にアクティブ状態にできるジョブの数 (MAXPTYn) (n=1 から 9)

出力待ち行列の管理

出力待ち行列は、ジョブの終了時にプリンター出力を管理するために役立ちます。出力待ち行列を効率的に保守して印刷出力を順調に処理するための方法を理解することは大切です。

プリンター出力は、出力待ち行列に入ります。出力待ち行列は、プリンター出力がプリンターによって処理される順序を判別します。出力待ち行列の管理によって、プリンター出力を円滑に処理できます。

関連概念

75 ページの『出力待ち行列』
出力待ち行列とは、プリンター出力ファイル (スプールされるファイルとも呼ばれる) が処理され、プリンターに送信されるのを待機する領域です。プリンター出力は、システム、または印刷ファイルを使用するユーザーのいずれかによって作成されます。

出力待ち行列の作成

出力待ち行列作成 (CRTOUTQ) コマンドは、スプール・ファイルの新規出力待ち行列を作成します。 項目は、それぞれのスプール・ファイルの出力待ち行列上に書き込まれます。ファイルが出力装置に書き込まれる順序は、スプール・ファイルの出力優先順位、および待ち行列プロンプト上のファイルの順序に指定された値 (SEQ パラメーター) によって決定されます。出力待ち行列の作成には、文字ベース・インターフェースを使用します。

コマンド: CRTOUTQ (出力待ち行列作成)

例: このコマンドは、DEPTAPRT という名前の出力待ち行列を作成し、それを現行ライブラリーに入れます。 AUT(*EXCLUDE) が指定され、OPRCTL(*YES) が想定されるので、出力待ち行列は、その待ち行列を作成したユーザーと、ジョブ制御権限またはスプール制御権限を持つユーザーだけが使用および制御できます。 SEQ(*FIFO) が指定されているので、スプール・ファイルは待ち行列上に先入れ先出し法の順序で入れられます。 Department A のユーザーがこの出力待ち行列の使用を許可される場合、オブジェクト権限付与 (GRTOBJAUT) コマンドを使用して、必要な権限を付与する必要があります。 この待ち行列上のファイルに含まれるデータは、ファイルの所有者であるユーザー、待ち行列の所有者、ジョブ制御権限を持つユーザー、またはスプール制御権限を持つユーザーだけが表示できます。デフォルトでは、それぞれのジョブの出力の先頭にはジョブ区切り文字は印刷されません。

```
CRTOUTQ   OUTQ(DEPTAPRT) AUT(*EXCLUDE) SEQ(*FIFO)
          TEXT('SPECIAL PRINTER FILES FOR DEPTA')
```

例: 以下は、出力待ち行列を作成する方法を示す別の例です。

```
CRTOUTQ   OUTQ(QGPL/JONES) +
          TEXT('Output queue for Mike Jones')
```

ジョブまたはジョブ記述への出力待ち行列の割り当て

新規に作成された出力待ち行列を使用する前に、それをジョブまたはジョブ記述に割り当てる必要があります。出力待ち行列は、System i ナビゲーター または文字ベース・インターフェースを使用して割り当てることができます。

System i ナビゲーター:

出力待ち行列をジョブに割り当てるために System i ナビゲーターを使用するには、以下のステップに従います。

1. System i ナビゲーターで、「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. ジョブを右クリックし、「**プロパティー (Properties)**」 → 「**プリンター出力**」をクリックします。

文字ベース・インターフェース:

新しい出力待ち行列を使用するようにジョブ記述を変更することもできます。したがって、ジョブ記述を使用するすべてのジョブは、新しい出力待ち行列を使用します。出力待ち行列をジョブ記述に割り当てるには、文字ベース・インターフェースを使用します。

コマンド: ジョブ記述変更 (CHGJOBD)

以下の例では、出力待ち行列 QPRINT を使用するようにジョブ記述 AMJOBS を変更します。

```
CHGJOBD JOBD(AMJOBS/AMJOBS) OUTQ(*LIBL/QPRINT)
```

プリンター出力へのアクセス

ジョブの実行が終了した後でプリンター出力を切り離す (プリンター出力をジョブから完全に分離する) ことができるので、System i ナビゲーター内のプリンター出力に基本操作または実行管理機能を介してアクセスすることができます。

System i ナビゲーター:

基本操作によってジョブのプリンター出力へアクセスするには、以下のようにします。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**基本操作**」 → 「**ジョブ**」の順に展開します。
2. プリンター出力を表示したいジョブを右マウス・ボタンでクリックし、「**プリンター出力**」をクリックします。「プリンター出力」ウィンドウが表示されます。

「出力待ち行列」フォルダーを介してプリンター出力にアクセスするには、以下のようにします。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**出力待ち行列**」の順に展開します。
2. プリンター出力を表示したい出力待ち行列 (たとえば、Qprint2) を選択します。出力待ち行列内のプリンター出力が表示されます。

文字ベース・インターフェース:

コマンド: 出力待ち行列処理 (WRKOUTQ <output queue name>)

コマンド: スプール・ファイル処理 (WRKSPLF JOB(qualified job name)

出力待ち行列の消去

ジョブがプリンター出力を作成すると、それは印刷のための出力待ち行列に送られます。ユーザーが、作成されたプリンター出力のすべてを実際に印刷することは、おそらくありません。 System i ナビゲーターでは、「**消去**」オプションによって出力待ち行列を消去することが可能です。出力待ち行列を消去すると、すべての出力が待ち行列から削除されます。

System i ナビゲーター:
出力待ち行列を消去するには、以下の手順で行います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (Connection)**」 → 「**実行管理機能**」 → 「**出力待ち行列**」の順に展開します。
2. 消去する出力待ち行列を右マウス・ボタン・クリックして、「**消去**」をクリックします。

文字ベース・インターフェース:
コマンド: 出力待ち行列消去 (CLROUTQ)

このコマンドは、印刷を待機しているかまたは保留中の出力待ち行列 QPRINT からのすべてのスプール・ファイルの項目を除去します。現在印刷中のファイルの項目、および現在実行中のプログラムからデータを受信中のファイルの項目は、影響を受けません。

```
CLROUTQ   OUTQ(QPRINT)
```

出力待ち行列の削除

出力待ち行列の削除には、文字ベース・インターフェースを使用することができます。

出力待ち行列を削除する前に、以下の要件を満たす必要があります。

削除される出力待ち行列には、項目を含めることはできません。それぞれのファイルの出力は、印刷、削除、または別の出力待ち行列への移動が必要です。サブシステムはアクティブにできません。待ち行列は、スプール書き出しプログラムの使用中としておくことはできません。待ち行列は、特定のプリンター向けのシステムが作成した場合は削除できません。

コマンド: 出力待ち行列削除 (DLTOUTQ)

このコマンドは、システムから出力待ち行列 PUNCH2 を削除します。

```
DLTOUTQ   OUTQ(PUNCH2)
```

システム上の出力待ち行列の表示

出力待ち行列は、プリンター出力がプリンターに送信される順序を決定します。出力待ち行列を表示するには、System i ナビゲーター を使用します。

システム上の出力待ち行列を表示するには、以下の指示に従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」の順に展開します。
2. 「**出力待ち行列**」をクリックします。

System i ナビゲーターでは、「組み込み」ウィンドウを使って、表示中の出力待ち行列のリストをカスタマイズすることができます。「組み込み」ウィンドウを使用すると、System i ナビゲーター に表示されるものを制限することができます。 たとえば、「組み込み」を実行して、特定の出力待ち行列だけを表示できます。

組み込み機能を使用するには、「表示」メニューの「**このビューのカスタマイズ**」をクリックします。

ジョブ・ログの管理

システム上のほとんどのジョブには、それに関連したジョブ・ログがあります。 ジョブ・ログには、いつジョブが開始したか、いつジョブが終了したか、どのコマンドが実行中か、障害通知およびエラー・メッセージなど、さまざまなことが記録されています。この情報を見ると、ジョブ・サイクルがどのように行われているかがよく分かります。

以下では、ジョブ・ログの処理時に実行できるさまざまなタスクについて説明します。

関連概念

84 ページの『ジョブ・ログ』
ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

ジョブ・ログ・サーバーの管理

QSYSWRK サブシステムは、ジョブ・ログ・サーバーを制御します。また幾つかのタスクを実行すると、ジョブ・ログ・サーバーをカスタマイズまたは管理することができます。

関連概念

84 ページの『ジョブ・ログ』
ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

ジョブ・ログ・サーバーの再構成:

出荷時には、ジョブ・ログ・サーバーは QSYSWRK で実行します。 QSYSWRK は絶えずアクティブです。パフォーマンスを強化するには、別のサブシステムでジョブ・ログ・サーバーが実行するように再構成することもできます。

別のサブシステムでジョブ・ログ・サーバーが実行するように再構成するには、文字ベース・インターフェースを使用し、以下のステップに従います。

1. QSYSWRK からのものと同一の経路指定項目を、サブシステム記述に追加する。 これは経路指定項目の、順序番号 500、プログラム QWCJLSVR、ライブラリー QSYS、比較値 'QJOBLOGSVR'、開始位置 1 です。
2. QJOBLOGSVR ジョブ記述内で指定されているジョブ待ち行列を、サブシステム上に存在しているジョブ待ち行列に変更する。
3. QJOBLOGAJ 自動開始ジョブ項目を (必要な場合は経路指定項目と共に) サブシステムに追加する。これにより、サブシステムの開始時にジョブ・ログ・サーバーは自動的に開始します。
   - または望む場合には、自動開始ジョブ項目を、始動プログラム内の STRLOGSVR コマンドへの呼び出しに置き換えることができます。
4. QJOBLOGAJ 自動開始ジョブ項目を QSYSWRK から除去します。

ジョブ・ログ・サーバーの別の再構成の例として、クラス変更 (CHGCLS) コマンドを使用して、QJOBLOGSVR クラス (ライブラリー QSYS 内) で指定された実行優先順位を変更できます。

```
CHGCLS CLS(QSYS/QJOBLOGSVR) RUNPTY(50)
```

関連概念

87 ページの『ジョブ・ログ・サーバー』
通常、ジョブ・ログ・サーバーはジョブのジョブ・ログをスプール・ファイルに書き込みます。ジョブ・ログを印刷装置または出力ファイルに経路指定できますが (QMHCTLJL、ジョブ・ログ制御 API を使用してそのように指定する場合)、ジョブ・ログを生成するにはこの方法は推奨されていません。

ジョブ・ログ・サーバーの終了:

ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドは、ジョブ・ログ・サーバーを終了するために使用します。 ジョブ・ログ・サーバーは、ジョブ・ログ保留状態にあるジョブのジョブ・ログを書き込みます。このコマンドの発行時に、複数のジョブ・ログ・サーバーのジョブがアクティブの場合、すべてのジョブ・ログ・サーバーのジョブが終了します。

このコマンドを使用するには、ジョブ制御 (*JOBCTL) 特殊権限を持っている必要があります。

重要: たとえば、処理時間が非常に長いまたは消費するリソースが多すぎるなどの理由で特定のジョブ・ログの生成を停止したいだけの場合は、関連トピック『***特定のジョブ・ログの作成の停止***』を参照してください。

ENDLOGSVR コマンドを使用する場合、サーバーを即時に終了するか (この方法はお勧めしません) または制御された方法で終了するかを指定できます。

関連概念

87 ページの『ジョブ・ログ・サーバー』
通常、ジョブ・ログ・サーバーはジョブのジョブ・ログをスプール・ファイルに書き込みます。ジョブ・ログを印刷装置または出力ファイルに経路指定できますが (QMHCTLJL、ジョブ・ログ制御 API を使用してそのように指定する場合)、ジョブ・ログを生成するにはこの方法は推奨されていません。

関連タスク

224 ページの『特定のジョブ・ログの作成の停止』
特定のジョブ・ログの作成のみを停止する場合は、ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドを使用しないでください。 ENDLOGSVR コマンドは、すべてのジョブ・ログ・サーバーを終了するため、すべてのジョブ・ログの作成が停止します。

228 ページの『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 →「**プリンター出力**」の下にあります。

System i ナビゲーター:

1. System i ナビゲーター から、ジョブ・ログ・サーバーが実行しているエンドポイント・システムを右マウス・ボタン・クリックし、「**コマンドの実行**」をクリックします。
2. 「実行するコマンド (Command to run)」で、ENDLOGSVR と入力します。
3. 「ジョブ・ログ・サーバーの終了 (End Job Log Server)」ウィンドウが表示され、このコマンドのパラメーターを指定できます。このウィンドウに必要なデータをすべて入力してから、「**OK**」をクリックします。 ウィンドウが閉じ、「コマンドの実行」ウィンドウに戻ります。
4. これで、「**OK**」をクリックしてコマンドを即時に実行するか、または「**スケジュール (Schedule)**」をクリックしてコマンド実行時刻をスケジュールするかのいずれかを実行できます。

文字ベース・インターフェース:
コマンド: ジョブ・ログ・サーバー終了 (ENDLOGSVR)

ジョブ・ログ・サーバーの開始

デフォルトで、ジョブ・ログ・サーバーは、QSYSWRK サブシステムが始動すると自動的に開始します。ジョブ・ログ・サーバー開始 (STRLOGSVR) コマンドを使用して、ジョブ・ログ・サーバーを手動で開始できます。

STRLOGSVR コマンドを使用すると、開始する追加のジョブ・ログ・サーバーの数を指定することも、必要な数をシステムに計算させることもできます。 要求されたサーバーの数が許容される最大アクティブ数を超えると、アクティブ・サーバーの最大数から現行数を引いた差の数だけが開始されます。アクティブにできる、またはジョブ待ち行列に同時に入れることができるジョブ・ログ・サーバーの最大数は 30 です。

関連概念

87 ページの『ジョブ・ログ・サーバー』
通常、ジョブ・ログ・サーバーはジョブのジョブ・ログをスプール・ファイルに書き込みます。ジョブ・ログを印刷装置または出力ファイルに経路指定できますが (QMHCTLJL、ジョブ・ログ制御 API を使用してそのように指定する場合)、ジョブ・ログを生成するにはこの方法は推奨されていません。

System i ナビゲーター:
System i ナビゲーターを使用するには、以下の指示に従います。

1. System i ナビゲーター で、ジョブ・ログ・サーバーがあるエンドポイント・システムを右マウス・ボタン・クリックして、「**コマンドの実行**」を選択します。
2. 「**実行するコマンド:**」フィールドに、STRLOGSVR と入力します。
3. 「**プロンプト**」をクリックします。
4. このコマンドのパラメーターを指定できる「ジョブ・ログ・サーバーの開始」ウィンドウが表示されます。このウィンドウに必要なデータをすべて入力してから、「**OK**」をクリックします。 ウィンドウが閉じ、「コマンドの実行」ウィンドウに戻ります。
5. これで、「**OK**」をクリックしてコマンドを即時に実行するか、または「**スケジュール (Schedule)**」をクリックしてコマンド実行時刻をスケジュールするかのいずれかを実行できます。

文字ベース・インターフェース:

コマンド: **ジョブ・ログ・サーバー開始** (STRLOGSVR)

ジョブ・ログの表示方法

実行管理機能の中でジョブにアクセスできる場所であれば、サブシステム・エリアまたはメモリー・プール・エリアなど、どこからでもジョブ・ログを参照することができます。ジョブ・ログを表示するには、System i ナビゲーターまたは文字ベース・インターフェースを使用できます。

関連タスク

229 ページの『ジョブ・ログ保留からのプリンター出力の生成』

System i ナビゲーター 「**Job Properties - Job Log (ジョブ・プロパティー - ジョブ・ログ)**」設定の「**Produce a job log (ジョブ・ログの生成)**」フィールドが選択されていないジョブでは、ジョブ・ログは生成されません。 その代わり、ジョブ・ログはジョブ・ログ保留状態になります。ジョブ・ログ保留状態にあるジョブ・ログからプリンター出力を生成するには、文字ベース・インターフェースを使用します。

228 ページの『ジョブ・ログ出力ファイルの削除』

ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 →「**プリンター出力**」の下にあります。

222 ページの『ジョブ・ログが表示されない場合に行うこと』

System i ナビゲーター でジョブ・ログを見つけて表示するには、ジョブ (バッチ・ジョブまたは対話式ジョブのいずれでも) を右マウス・ボタン・クリックして、メニューから「**ジョブ・ログ**」をクリックします。 ただし、ジョブの状況またはジョブ・ログ値がジョブ記述でどのように設定されているかによって、ジョブ・ログは出力待ち行列にあるか、ジョブ・ログ保留状況になっている場合があり、ジョブ・ログを使用できない場合もあります。

180 ページの『サブシステムの停止』

System i ナビゲーター または文字ベース・インターフェースを使用して、1 つまたは複数のアクティブ・サブシステムを停止して、処理されているアクティブ作業をどうするか指定することができます。サブシステムが停止すると、そのサブシステムでは新しいジョブまたは経路指定ステップは開始されません。

関連情報

印刷の管理

ジョブ・システム値: 即時終了の最大時間

System i ナビゲーター:

アクティブ・ジョブまたはサーバー・ジョブのジョブ・ログにアクセスするには、以下のようにします。

1. System i ナビゲーターで、**「ユーザー接続」** → **「接続 (connection)」** → **「実行管理機能」** → **「アクティブ・ジョブ」**または**「サーバー・ジョブ」**の順に展開します。
2. ジョブ (たとえば、Qbatch) を右マウス・ボタン・クリックして、**「ジョブ・ログ」**をクリックします。詳細については、「ジョブ・ログ」ウィンドウでヘルプを開いて参照してください。

   メッセージの詳細を表示する場合は、メッセージを右マウス・ボタン・クリックして、**「プロパティー」**をクリックします。「メッセージ・プロパティー」ウィンドウには、詳細なメッセージ情報が表示されます。このウィンドウには、メッセージの詳細とともにメッセージ・ヘルプが示されます。詳細なメッセージ・ヘルプは、問題を解決するための情報を提供します。

以下のリストでは、ジョブ・ログにアクセスするための追加の方法を記載しています。

- **「基本操作」** → **「プリンター (Printer)」**
- **「基本操作」** → **「ジョブ」** → **ジョブを右マウス・ボタン・クリック** → **「プリンター出力」**
- **「実行管理機能」** → **「アクティブ・ジョブ」** → **ジョブを右マウス・ボタン・クリック** → **「プリンター出力」**
- **「実行管理機能」** → **「出力待ち行列」**
- **「ユーザーおよびグループ (Users and Groups)」** → **「すべてのユーザー (All Users)」** → **ユーザーを右マウス・ボタン・クリック** → **「ユーザー・オブジェクト (User Objects)」** → **「プリンター出力」**

文字ベース・インターフェース:

ジョブ・ログを表示する方法は、ジョブの状況によって異なります。

- 完了ジョブの保留ジョブ・ログ、すべてのジョブ・ログのスプール・ファイル、あるいはその両方を表示する場合は、**ジョブ・ログ処理 (WRKJOBLOG)** コマンドを使用できます。例えば、終了したすべてのジョブの保留ジョブ・ログのリストを表示するには、次のコマンドを使用します。

  ```
  WRKJOBLOG JOBLOGSTT(*PENDING)
  ```

- ジョブがまだアクティブの状態にある (バッチ・ジョブまたは対話式ジョブ) かまたはジョブ待ち行列にあってまだ開始されていない場合は、**ジョブ・ログ表示 (DSPJOBLOG)** コマンドを使用します。例えば、ユーザー JSMITH の対話式ジョブのジョブ・ログをディスプレイ装置 WS1 に表示するには、次のコマンドを使用します。

  ```
  DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
  ```

  nnnnnn はジョブ番号です。

- ジョブが終了し、ジョブ・ログがまだ印刷されていない場合は、**スプール・ファイル表示 (DSPSPLF)** コマンドを使用します。例えば、ユーザー FRED に関連付けられているジョブ番号 001293 のジョブ・ログをディスプレイ装置 WS3 に表示するには、次のコマンドを使用します。

  ```
  DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
  ```

上記のコマンドの使用について十分な情報をお持ちでない場合は、ユーザー・ジョブ処理 (WRKUSRJOB) コマンドまたは実行依頼済みジョブ処理 (WRKSBMJOB) コマンドが役立つ場合があります。

ジョブ・ログが表示されない場合に行うこと

System i ナビゲーター でジョブ・ログを見つけて表示するには、ジョブ (バッチ・ジョブまたは対話式ジョブのいずれでも) を右マウス・ボタン・クリックして、メニューから**「ジョブ・ログ」**をクリックしま

す。 ただし、ジョブの状況またはジョブ・ログ値がジョブ記述でどのように設定されているかによって、ジョブ・ログは出力待ち行列にあるか、ジョブ・ログ保留状況になっている場合があり、ジョブ・ログを使用できない場合もあります。

以下は、ジョブでジョブ・ログ・メニュー・オプションを使用できない場合に取るべきステップです。

ヒント: アクティブ・ジョブ (またはサーバー・ジョブ) の列表示を「状況」を組み込むよう設定します。これによって、ジョブ・ログをどこで探すかを迅速に判別できるようになります。

次のようにしてジョブ・ログにアクセスします。**「ユーザー接続」 → *ご使用のシステム* → 「実行管理機能」 → 「アクティブ・ジョブ」または「サーバー・ジョブ」 → ジョブを右クリックしてジョブ・ログを選択する。**

ジョブ・ロブ・メニュー・オプションを使用できない場合、またはシステムがジョブ・ログを検索できないことを伝えるエラー・メッセージが表示される場合は、以下を考慮してください。

1. ジョブの状況を確認する。

オプション	説明
実行中	「ジョブのプロパティー」-「ジョブ・ログ」ウィンドウを確認して、**「ジョブ・ログの作成」**ボックスにチェック・マークが付いていることを確認します。チェック・マークが付いていない場合は、ジョブ・ログは作成されていません。
終了済み	このジョブは正常な方法で終了しませんでした。エラーまたはユーザー介入が原因として考えられます。ジョブを右マウス・ボタンでクリックして、**「印刷装置出力」**をクリックします。 ジョブ・ログが表示されていない場合は、「ジョブのプロパティー」-「ジョブ・ログ」ウィンドウを確認して、「ジョブ・ログのプリンター出力を作成 (Produce printer output for job log)」チェック・ボックスが選択されていることを確認します。
完了 - プリンター出力は使用可能	このジョブは正常に終了しました。ジョブを右マウス・ボタンでクリックして、**「印刷装置出力」**をクリックします。 ジョブ・ログが表示されていない場合は、「ジョブのプロパティー」-「ジョブ・ログ」ウィンドウを確認して、**「ジョブが正常に終了した場合にジョブ・ログのプリンター出力を作成」**フィールドにチェック・マークが付いていることを確認します。
完了 - ジョブ・ログ保留	ジョブ・ログは作成されません。 ジョブ・ログは除去されるまで保留のままです。保留ジョブ・ログを表示するには、ジョブ・ログ表示 (DSPJOBLOG) コマンドを使用する必要があります。

2. ジョブ・ログは出力待ち行列にスプールされて印刷されている可能性がある。その場合、ログはシステムから除去されています。
3. ジョブ・ログは別のユーザーによって削除された可能性がある。

関連タスク

221 ページの『ジョブ・ログの表示方法』
実行管理機能の中でジョブにアクセスできる場所であれば、サブシステム・エリアまたはメモリー・プ

ール・エリアなど、どこからでもジョブ・ログを参照することができます。ジョブ・ログを表示するには、System i ナビゲーターまたは文字ベース・インターフェースを使用できます。

ジョブ・ログ用の出力待ち行列の指定

デフォルトでは、ジョブ・ログをスプールするのに使用されるプリンター・ファイルは QPJOBLOG です。システム上に、複数の QPJOBLOG プリンター・ファイルを持つことができます。QSYS では、OUTQ 属性を使用する出力待ち行列は QUSRSYS ライブラリー内にある QEZJOBLOG です。システムがジョブ・ログを作成すると、ジョブのライブラリー・リスト内で QPJOBLOG プリンター・ファイルを検索します。最初に見つかるファイルが使用されます。こうした設定を調整するには、文字ベース・インターフェースを使用します。

1. プリンター・ファイル QPJOBLOG OUTQ 属性を *JOB に変更します。
   a. **コマンド**: 印刷装置ファイル変更 `CHGPRTF FILE(QPJOBLOG) OUTQ(*JOB)`
2. 希望する出力待ち行列にジョブの OUTQ 属性を変更します。 文字ベース・インターフェースまたは System i ナビゲーター を使用すると、これを実行できます。
   a. コマンド: ジョブ変更 `CHGJOB OUTQ(MYLIB/MYOUTQ)`
   b. System i ナビゲーター: **「実行管理機能」 → 「アクティブ・ジョブ」 → ジョブを右マウス・ボタンでクリックして「プロパティー」を選択する → 「Printer Tab (印刷装置タブ)」**

関連情報

出力待ち行列またはプリンターへの印刷の制御

特定のジョブ・ログの作成の停止

特定のジョブ・ログの作成のみを停止する場合は、ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドを使用しないでください。 ENDLOGSVR コマンドは、すべてのジョブ・ログ・サーバーを終了するため、すべてのジョブ・ログの作成が停止します。

特定のジョブ・ログの作成を停止するには、以下の手順を使用します。

1. System i ナビゲーター で、ジョブ・ログの作成を停止するジョブを右クリックして、**「プロパティー」**をクリックします。 (**「ユーザー接続」 → 「接続 (connection)」 → 「実行管理機能」 → 「アクティブ・ジョブ」または「サーバー・ジョブ」**)
2. **「ジョブ・ログ」**タブをクリックします。
3. **「ジョブ・ログの作成」**ボックスのチェック・マークを外して、**「OK」**をクリックします。

ジョブ・ログの作成が停止し、ジョブ・ログはジョブ・ログ保留状況になります。

関連概念

85 ページの『ジョブ・ログの作成方法』
ジョブ・ログは必要なときに使用できますが、ジョブ・ログの必要がない場合はそれを生成する作業は実行されません。

関連タスク

219 ページの『ジョブ・ログ・サーバーの終了』
ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドは、ジョブ・ログ・サーバーを終了するために使用します。 ジョブ・ログ・サーバーは、ジョブ・ログ保留状態にあるジョブのジョブ・ログを書き込みます。このコマンドの発行時に、複数のジョブ・ログ・サーバーのジョブがアクティブの場合、すべてのジョブ・ログ・サーバーのジョブが終了します。

228 ページの『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API ある

いはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 → 「**プリンター出力**」の下にあります。

227 ページの『バッチ・ジョブのログ情報の制御』
バッチ・アプリケーションの場合に、ログに記録される情報量を変更できます。 IBM 提供のサブシステム QBATCH のジョブ記述で指定されるログ・レベル (LOG(40 *NOLIST)) は、ジョブが異常終了した場合に完全なログを提供します。ジョブが正常に完了した場合、ジョブ・ログは生成されません。

ジョブ・ログの作成の抑制

ジョブ・ログの作成の抑制は、ジョブ・ログが不要であることが分かっていて、システム・リソースを節約する場合に役立ちます。ジョブ・ログを作成しないことを指定すると、ジョブ・ログは生成されなくなり、保留ジョブ・ログ除去 (QWTRMVJL) コマンドまたはジョブ終了 (ENDJOB) コマンドで除去しない限り、保留のままになります。

ジョブ・ログの作成を抑制するには、以下の指示に従います。

1. System i ナビゲーター で、「**ジョブのプロパティー**」-「**ジョブ・ログ**」ウィンドウを開きます。 (「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」(**または**「**サーバー・ジョブ**」) → **ジョブを右クリック** → 「**プロパティー**」 → 「**ジョブ・ログ**」**タブ**)
2. 「**ジョブ・ログの作成**」ボックスのチェック・マークを外して、「**OK**」をクリックします。

関連概念

85 ページの『ジョブ・ログの作成方法』
ジョブ・ログは必要なときに使用できますが、ジョブ・ログの必要がない場合はそれを生成する作業は実行されません。

関連タスク

228 ページの『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 → 「**プリンター出力**」の下にあります。

227 ページの『バッチ・ジョブのログ情報の制御』
バッチ・アプリケーションの場合に、ログに記録される情報量を変更できます。 IBM 提供のサブシステム QBATCH のジョブ記述で指定されるログ・レベル (LOG(40 *NOLIST)) は、ジョブが異常終了した場合に完全なログを提供します。ジョブが正常に完了した場合、ジョブ・ログは生成されません。

ジョブ・ログ内の情報の制御

問題を処理するときに、頻繁に問題が生じるジョブについては記録される情報量を最大にしたい場合があります。別の状況として、正常に完了したジョブについてはジョブ・ログを作成したくない場合もあります。あるいは、通知メッセージは除外したいという場合もあります。

ジョブ・ログにどのような情報を追加するかは、ジョブ記述でメッセージ・レベル、メッセージ重大度、またはメッセージ・テキスト・レベルの値を設定することで制御できます。ただし、特定のジョブのジョブ・ログに書き込む情報を制御したい場合は、System i ナビゲーター 内の「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」 ウィンドウを使用します。

このウィンドウにより、以下を制御できます。

- ジョブ・ログを生成するかどうか、および生成に使用する方式
- 最大サイズに達したときの対応
- 制御言語プログラムからのコマンドをログに記録するかどうか
- ジョブ・ログにメッセージを保持するかどうか、およびどの特定のメッセージを保持するか (ログ・レベルおよびメッセージ重大度)
- ジョブが正常に終了した場合にジョブ・ログのプリンター出力を作成するかどうか、および印刷する内容

「ジョブ・プロパティー (Job Properties)」-「ジョブ・ログ (Job Log)」ウィンドウにアクセスするには、以下のステップに従います。

1. System i ナビゲーター から、ジョブの「ジョブ・プロパティー (Job Properties)」ウィンドウを開き、「**ジョブ・ログ (Job Log)**」タブをクリックします。 「**ユーザー接続**」 → 「**接続 (Connection)**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」 → **ジョブを右クリック** → 「**プロパティー**」の順に展開します。
2. このウィンドウ上で使用できるさまざまなオプションの詳細説明については、オンライン・ヘルプを参照してください。

関連概念

85 ページの『ジョブ・ログの作成方法』
ジョブ・ログは必要なときに使用できますが、ジョブ・ログの必要がない場合はそれを生成する作業は実行されません。

関連タスク

229 ページの『保留ジョブ・ログのクリーンアップ』
保留ジョブ・ログからジョブをクリーンアップまたは除去するにはいくつかの方法があります。ジョブは、最大ログ項目 (LOGLMT) パラメーターの値を 0 に設定して終了できます。 ジョブがすでに終了している場合、保留ジョブ・ログ除去 (QWTRMVJL) API を実行できます。ジョブ・ログ処理 (WRKJOBLOG) コマンドも使用できます。

228 ページの『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、「**基本操作**」 →「**プリンター出力**」の下にあります。

ジョブのログ・レベルの変更:

ジョブのログ・レベルは、ログに記録されるメッセージ・タイプの特定の組み合わせに割り当てられる数値レベルです。ジョブ記述内のログ・レベルは、文字ベース・インターフェースを使用して変更できます。ただし、特定のジョブのログ・レベルを変更したい場合は、System i ナビゲーター 内の「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」 ウィンドウを使用します。

「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」ウィンドウにアクセスするには、以下のステップに従います。

1. System i ナビゲーターで、「**ユーザー接続**」 → 「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. ジョブを選択し、「**プロパティー (Properties)**」を右クリックします。

3. 特定のジョブのプロパティー・ウィンドウで、「**ジョブ・ログ**」タブを選択し、ロギング・レベルを変更します。

関連概念

89 ページの『メッセージ』
メッセージには、ジョブ名、メッセージ・タイプ、送信された日時、発生したアクション、および問題を修正するのに必要なアクションが含まれています。これは、サーバーで生じる問題をトラブルシューティングしようとする際に役立ちます。System i ナビゲーター によって、サーバー・ジョブのジョブ・ログにアクセスするすることができます。 メッセージは、警告メッセージとジョブ・ログに記録されたメッセージという 2 つのカテゴリーに分類されます。

84 ページの『ジョブ・ログ』
ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

関連タスク

229 ページの『保留ジョブ・ログのクリーンアップ』
保留ジョブ・ログからジョブをクリーンアップまたは除去するにはいくつかの方法があります。ジョブは、最大ログ項目 (LOGLMT) パラメーターの値を 0 に設定して終了できます。 ジョブがすでに終了している場合、保留ジョブ・ログ除去 (QWTRMVJL) API を実行できます。ジョブ・ログ処理 (WRKJOBLOG) コマンドも使用できます。

バッチ・ジョブのログ情報の制御:

バッチ・アプリケーションの場合に、ログに記録される情報量を変更できます。 IBM 提供のサブシステム QBATCH のジョブ記述で指定されるログ・レベル (LOG(40 *NOLIST)) は、ジョブが異常終了した場合に完全なログを提供します。ジョブが正常に完了した場合、ジョブ・ログは生成されません。

ジョブ待ち行列レベル (QBATCH) でのジョブ・ログの制御は、QBATCH サブシステム・ジョブのジョブ・ログ設定を調整することで実行されます。サブシステムのジョブ・レベルでのジョブ・ログの生成方法を制御するためのオプションとして、個人のジョブ・レベルで実行する場合のものと同じオプションがあります。

ジョブ待ち行列サブシステムのジョブ・ログ設定を調整するには、以下のようにします。

System i ナビゲーター で、ジョブ待ち行列サブシステムの「**サブシステム・プロパティー (Subsystem Properties)**」-「**ジョブ・ログ**」ウィンドウを開きます。 (「**実行管理機能**」 → 「**サブシステム**」 → 「**アクティブ・サブシステム**」 → 「**QBATCH**」 → **QBATCH ジョブを右マウス・ボタン・クリック** → 「**プロパティー**」 → 「**ジョブ・ログ**」**タブ**)

注: サブシステムの「**ジョブ・ログ・フィールドの生成 (Produce a job log field)**」フィールド (*PND) のチェック・マークを外している場合、サブシステムに固有のジョブ・ログは、他のプリンター出力と共にはリストされません。 保留状態のジョブ・ログを表示するには、ジョブ・ログ表示 (DSPJOBLOG) コマンドを使用する必要があります。

バッチ・ジョブが制御言語プログラムを実行している場合、制御言語プログラム・コマンドは、LOGCLPGM(*YES) が制御言語プログラム作成 (CRTCLPGM) コマンドまたはプログラム変更 (CHGPGM) コマンド上に指定されている場合のみ、ログに記録されます。

関連概念

84 ページの『ジョブ・ログ』
ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

関連タスク

『ジョブ・ログ出力ファイルの削除』
ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、**「基本操作」 →「プリンター出力」**の下にあります。

224 ページの『特定のジョブ・ログの作成の停止』
特定のジョブ・ログの作成のみを停止する場合は、ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドを使用しないでください。 ENDLOGSVR コマンドは、すべてのジョブ・ログ・サーバーを終了するため、すべてのジョブ・ログの作成が停止します。

225 ページの『ジョブ・ログの作成の抑制』
ジョブ・ログの作成の抑制は、ジョブ・ログが不要であることが分かっていて、システム・リソースを節約する場合に役立ちます。ジョブ・ログを作成しないことを指定すると、ジョブ・ログは生成されなくなり、保留ジョブ・ログ除去 (QWTRMVJL) コマンドまたはジョブ終了 (ENDJOB) コマンドで除去しない限り、保留のままになります。

ジョブ・ログ出力ファイルの削除

ジョブ・ログは、ジョブが正常に完了した場合、または保留ジョブ・ログ除去 (QWTRMVJL) API あるいはジョブ終了 (ENDJOB) コマンドの発行時にシステムから除去されます。 さらに、「未完了ジョブ・ログのクリア (clear incomplete job logs)」を IPL 時に指定した場合、保留ジョブ・ログ内のすべてのジョブは、IPL 時にシステムから除去されます。 残っているすべてのジョブ・ログ出力ファイルは、**「基本操作」 →「プリンター出力」**の下にあります。

「プリンター出力」にあるジョブ・ログを削除するには、削除するジョブ・ログのファイル名を右マウス・ボタン・クリックして、**「削除」**をクリックします。

ジョブ・ログの削除が安全かどうかを判別する方法

ジョブ・ログを保持するかまたは削除するかの決定のバランスを取ることは、挑戦になります。ジョブ・ログは、問題をトラブルシューティングできるように保持しておく必要があるものです。ジョブ・ログは、システムが乱雑になるので、保持しておきたくないものでもあります。どのジョブ・ログを削除するか、またはどのジョブ・ログを生成しないようにするかを決定する場合、以下の指針を考慮してください。

- これはジョブ・ログを見なくても簡単に修正できるジョブか ?
- これはシステム内の他のジョブと類似のジョブか ? これが失敗する場合、類似のジョブも失敗する可能性があるか ? その場合、1 つのジョブだけにジョブ・ログを生成させることができます。

関連概念

84 ページの『ジョブ・ログ』
ジョブ・ログには、ジョブに入れられた要求に関連する情報が含まれます。ジョブ・ログには、保留形式とスプール形式の 2 つの形式があります。

関連タスク

221 ページの『ジョブ・ログの表示方法』
実行管理機能の中でジョブにアクセスできる場所であれば、サブシステム・エリアまたはメモリー・プール・エリアなど、どこからでもジョブ・ログを参照することができます。ジョブ・ログを表示するには、System i ナビゲーターまたは文字ベース・インターフェースを使用できます。

219 ページの『ジョブ・ログ・サーバーの終了』
ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドは、ジョブ・ログ・サーバーを終了するために

使用します。 ジョブ・ログ・サーバーは、ジョブ・ログ保留状態にあるジョブのジョブ・ログを書き込みます。このコマンドの発行時に、複数のジョブ・ログ・サーバーのジョブがアクティブの場合、すべてのジョブ・ログ・サーバーのジョブが終了します。

224 ページの『特定のジョブ・ログの作成の停止』
特定のジョブ・ログの作成のみを停止する場合は、ジョブ・ログ・サーバー終了 (ENDLOGSVR) コマンドを使用しないでください。 ENDLOGSVR コマンドは、すべてのジョブ・ログ・サーバーを終了するため、すべてのジョブ・ログの作成が停止します。

225 ページの『ジョブ・ログの作成の抑制』
ジョブ・ログの作成の抑制は、ジョブ・ログが不要であることが分かっていて、システム・リソースを節約する場合に役立ちます。ジョブ・ログを作成しないことを指定すると、ジョブ・ログは生成されなくなり、保留ジョブ・ログ除去 (QWTRMVJL) コマンドまたはジョブ終了 (ENDJOB) コマンドで除去しない限り、保留のままになります。

225 ページの『ジョブ・ログ内の情報の制御』
問題を処理するときに、頻繁に問題が生じるジョブについては記録される情報量を最大にしたい場合があります。別の状況として、正常に完了したジョブについてはジョブ・ログを作成したくない場合もあります。あるいは、通知メッセージは除外したいという場合もあります。

227 ページの『バッチ・ジョブのログ情報の制御』
バッチ・アプリケーションの場合に、ログに記録される情報量を変更できます。 IBM 提供のサブシステム QBATCH のジョブ記述で指定されるログ・レベル (LOG(40 *NOLIST)) は、ジョブが異常終了した場合に完全なログを提供します。ジョブが正常に完了した場合、ジョブ・ログは生成されません。

ジョブ・ログ保留からのプリンター出力の生成

System i ナビゲーター 「**Job Properties - Job Log (ジョブ・プロパティー - ジョブ・ログ)**」設定の「**Produce a job log (ジョブ・ログの生成)**」フィールドが選択されていないジョブでは、ジョブ・ログは生成されません。 その代わり、ジョブ・ログはジョブ・ログ保留状態になります。ジョブ・ログ保留状態にあるジョブ・ログからプリンター出力を生成するには、文字ベース・インターフェースを使用します。

コマンド: ジョブ・ログ表示 (DSPJOBLOG)

関連概念

87 ページの『ジョブ・ログ保留』
ジョブ・ログ保留状態は、これまで何年も使用されてきました。 ジョブのジョブ・ログ属性が *PND の場合、ジョブ・ログは生成されません。特定のジョブのジョブ・ログをどのように、またどのような状況下で生成するかを制御することができます。

関連タスク

221 ページの『ジョブ・ログの表示方法』
実行管理機能の中でジョブにアクセスできる場所であれば、サブシステム・エリアまたはメモリー・プール・エリアなど、どこからでもジョブ・ログを参照することができます。ジョブ・ログを表示するには、System i ナビゲーターまたは文字ベース・インターフェースを使用できます。

保留ジョブ・ログのクリーンアップ

保留ジョブ・ログからジョブをクリーンアップまたは除去するにはいくつかの方法があります。ジョブは、最大ログ項目 (LOGLMT) パラメーターの値を 0 に設定して終了できます。 ジョブがすでに終了している場合、保留ジョブ・ログ除去 (QWTRMVJL) API を実行できます。ジョブ・ログ処理 (WRKJOBLOG) コマンドも使用できます。

LOGMLT を 0 に設定してジョブを終了するには、System i ナビゲーター または文字ベース・インターフェースを使用します。

関連概念

87 ページの『ジョブ・ログ保留』
ジョブ・ログ保留状態は、これまで何年も使用されてきました。 ジョブのジョブ・ログ属性が *PND の場合、ジョブ・ログは生成されません。特定のジョブのジョブ・ログをどのように、またどのような状況下で生成するかを制御することができます。

関連タスク

225 ページの『ジョブ・ログ内の情報の制御』
問題を処理するときに、頻繁に問題が生じるジョブについては記録される情報量を最大にしたい場合があります。別の状況として、正常に完了したジョブについてはジョブ・ログを作成したくない場合もあります。あるいは、通知メッセージは除外したいという場合もあります。

226 ページの『ジョブのログ・レベルの変更』
ジョブのログ・レベルは、ログに記録されるメッセージ・タイプの特定の組み合わせに割り当てられる数値レベルです。ジョブ記述内のログ・レベルは、文字ベース・インターフェースを使用して変更できます。ただし、特定のジョブのログ・レベルを変更したい場合は、System i ナビゲーター 内の「**ジョブ・プロパティー (Job Properties)**」-「**ジョブ・ログ (Job Log)**」 ウィンドウを使用します。

関連情報

終結処置の変更 (CHGCLNUP) コマンド

Exit Program for Tailoring Automatic Cleanup

System i ナビゲーター:

1. System i ナビゲーターで、「**実行管理機能**」 → 「**アクティブ・ジョブ**」の順に展開します。
2. 終了するジョブを見つけます。
3. ジョブを右マウス・ボタンでクリックして、「**削除/終了**」をクリックします。
4. 「削除/終了の確認 (Confirm Delete/End)」ウィンドウで、「**プリンター出力の削除 (Delete printer output)**」を「いいえ (No)」に設定します。
5. 「削除/終了の確認 (Confirm Delete/End)」ウィンドウを完成させて、「**削除**」をクリックします。

文字ベース・インターフェース:
コマンド:ジョブ終了 (ENDJOB LOGLMT(0))

ジョブ会計の管理

ジョブ会計機能は、デフォルトではアクティブ状態になっていません。この機能をセットアップするには、最初に幾つかのステップが必要です。以下の情報では、ジョブ会計をセットアップして、ジョブ会計に関連する最も一般的なタスクの幾つかを実行する方法が取り上げられています。

関連概念

94 ページの『ジョブ会計』
ジョブ会計機能はデータを収集し、システムの使用者および使用しているシステム・リソースを判別することができます。さらに、システム全体の使用を評価するのにも役立ちます。ジョブ会計は、オプションです。ジョブ会計をセットアップするには、特定のステップを行う必要があります。システムに対して、ジョブ・リソースの会計データ、プリンター・ファイルの会計データ、またはその両方を収集するように要求できます。さらに、ユーザー・プロファイルまたは特定のジョブに対して、会計コードを割り当てることもできます。

関連情報

ジャーナル管理

ジャーナル処理のセットアップ

ジョブ会計のセットアップ

ジョブ会計をセットアップするには、文字ベース・インターフェースを使用します。

1. ジャーナル・レシーバーを作成します。ジャーナル・レシーバーは、任意の名前と選択するライブラリーで作成できます。ACGJRN1 などの命名規則で名前を付けることが推奨されています。そのようにするなら、ジャーナル変更 CHGJRN JRNRCV(*GEN) コマンドを使用して追加のレシーバー (ACGJRN2、ACGJRN3 など) を作成できます。

   a. **コマンド**: ジャーナル・レシーバー作成 (CRTJRNRCV)

   ```
   CRTJRNRCV JRNRCV(USERLIB/ACGJRN1)
   ```

2. ジョブ会計ジャーナルを作成します。ジャーナル名は QSYS/QACGJRN でなければならず、 QSYS ライブラリーにオブジェクトを追加する権限が必要です。

   a. **コマンド**: ジャーナル作成 (CRTJRN)

   ```
   CRTJRN JRN(QSYS/QACGJRN) JRNRCV(USERLIB/ACGJRN1) AUT(*EXCLUDE)
   ```

   このジャーナル・レシーバーは、ステップ 1 で作成したレシーバーと同じでなければなりません。任意の選択対象に権限を設定できますが、収集されたデータを使用してリソースの使用に関してユーザーに課金できるようにするため *EXCLUDE をお勧めします。

3. ジャーナル会計情報 (QACGLVL) システム値を変更します。このシステム値は、ジャーナル・ジョブ会計情報またはプリンター情報、あるいはその両方に設定できます。*JOB を使用するとジョブ (JB) ジャーナル項目が生成され、*PRINT を使用すると直接印刷 (DP) またはスプール印刷 (SP) ジャーナル項目が生成されます。 値 *NONE は、ジャーナル QACGJRN でジャーナル処理が行われないことを意味します。ジョブ会計データは、システム値が *NONE 以外の値に設定された後に開始されたジョブでのみジャーナル処理が行われます。

   a. **コマンド**: システム値の処理 (WRKSYSVAL) またはシステム値変更 (CHGSYSVAL)

   ```
   CHGSYSVAL SYSVAL(QACGLVL) VALUE('*JOB *PRINT')
   ```

4. 各ユーザー・プロファイルごとに、会計コード・パラメーター ACGCDE を設定します。会計コードには、長さ 15 文字以内の任意の英数字文字列を設定できます。ジョブ会計ジャーナル項目の分析において現行ユーザーが重要であることが分かる場合、ACGCDE パラメーターをユーザー・プロファイルの名前に設定することをお勧めします。

   a. **コマンド**: ユーザー・プロファイル変更 (CHGUSRPRF) またはユーザー・プロファイル作成 (CRTUSRPRF)

   ```
   CHGUSRPRF USRPRF(USERID1) ACGCDE(USERID1)
   ```

   またジョブ記述変更 (CHGJOBD) コマンドまたはジョブ記述作成 (CRTJOBD) コマンドを使用して、ユーザーのグループに会計コードを指定することもできます。

   ジョブ記述のデフォルトの会計コードは *USRPRF で、ジョブのユーザー・プロファイルの会計コードを使用することを意味します。*USRPRF 以外の値をジョブ記述で指定すると、ユーザー・プロファイルで指定される会計コードよりも優先順位が高くなります。

関連概念

99 ページの『会計コードについて』
ジョブの初期会計コード (長さは最大 15 文字) は、ジョブ記述またはジョブのユーザー・プロファイル内の ACGCDE (会計コード) パラメーターによって決定されます。

アカウンティング・コードの割り当ての制御

すべてのデータ処理アプリケーションにとって重要な側面は、必ず正しい制御フィールドを指定することです。ジョブ・アカウンティング・コードの場合、これには正式なコードの存在を検査するだけでなく、どのユーザーが特定コードの使用を許可されているかをも検査する、複合の妥当性検査機能が必要となることがあります。

アカウンティング・コードは、以下の領域で割り当て可能です。

- ユーザー・プロファイル
- ジョブ記述
- ジョブ内 (アカウンティング・コード変更 (CHGACGCDE) コマンド)

アカウンティング・コードの割り当ての制御が重要である場合は、以下を考慮してください。

1. アカウンティング・コードをユーザー・プロファイルに入れる前に、そのコードが特定のユーザーに対して有効であることを確認します。
2. 機密保護担当者権限だけを CHGACGCDE コマンドに付与することで、ジョブ記述変更 (CHGJOBD) コマンド上でのアカウンティング・コードの変更を制御します。
   - または、CHGACGCDE コマンドを使用して、ユーザーに自身のまたは他のジョブのジョブ・アカウンティング・コードの変更を許可します。 他のジョブを変更する場合は、ユーザーは特殊権限 *JOBCTL も持っている必要があります。
3. ジョブ待ち行列上のジョブのアカウンティング・コードの変更、またはあるジョブから別のジョブのアカウンティング・コードへの変更を回避するには、制御言語プログラムおよびコマンドを使用します。たとえば、CHGACGCDE コマンドは私用レベルで権限を与えて、制御言語プログラムに組み込み、現行ジョブ (JOB(*) などが指定されている場合) だけを変更できます。 コマンドは適切な権限を持ちます。

関連概念

98 ページの『セキュリティーおよびジョブ会計』
機密保護担当者 (またはこの権限を与えられているプログラム) か *ALLOBJ および *SECADM 権限を持つユーザーのみが、ジャーナル会計情報 (QACGLVL) システム値を変更できます。

99 ページの『会計コードについて』
ジョブの初期会計コード (長さは最大 15 文字) は、ジョブ記述またはジョブのユーザー・プロファイル内の ACGCDE (会計コード) パラメーターによって決定されます。

収集されたデータの表示

ジョブ会計ジャーナルにデータを収集した後、ジャーナル項目をファイルに書き込んで表示することができます。

このためには、以下のステップを実行します。

注: 次の例では、ジョブ会計ジャーナル名は QACGJRN です。

1. システム提供のモデル出力ファイルのコピーを会計ジャーナル用に作成します。QAJBACG4 は、*TYPE4 出力ファイル形式のモデル出力ファイルです。
   a. **コマンド**: 複製オブジェクト作成 (CRTDUPOBJ)

```
CRTDUPOBJ OBJ(QAJBACG4) FROMLIB(QSYS) OBJTYPE(*FILE) TOLIB(QTEMP)
   NEWOBJ(MYJBACG4)
```

2. 作成した出力ファイルにジャーナル項目をダンプします。次の例では、JB またはジョブ・タイプ・ジャーナル項目のみがダンプされます。

a. **コマンド**: ジャーナル表示 (DSPJRN)

```
DSPJRN JRN(QACGJRN) ENTTYP(JB) OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4)
   OUTFILE(QTEMP/MYJBACG4)
```

3. SQL セッションを開始します。その後、SQL セッションで SELECT コマンドを使用して、表示するフィールドを選択します。

   a. **コマンド**: 構造化照会言語の開始 (STRSQL)

```
STRSQL
SELECT JAJOB, JAUSER, JAUSPF, JACDE, JACPU FROM QTEMP/MYJBACG4
```

対話的にフィールド名のリストを表示するか、QUERY 処理 (WRKQRY) コマンドを使用して QUERY を作成および実行してファイルに出力します。

ジョブ会計ジャーナル項目の変換

ジャーナル表示 (DSPJRN) コマンドで OUTFILE パラメーターを使用して、ジョブ会計ジャーナル項目を処理可能なデータベース・ファイルに書き出すことができます。

OUTFILE パラメーターを使用すると、ファイルまたはメンバーに名前を付けることができます。 メンバーが存在する場合には、レコードが書き込まれる前に消去されます。メンバーが存在しないと、追加されます。ファイルが存在しない場合、レコード様式 QJORDJE を使用してファイルが作成されます。この様式では、各ジャーナル項目の標準ヘッディング・フィールドが定義されますが、ジョブ会計データは単一の大規模なフィールドとして定義されます。

会計データを単一の大規模なフィールドとして処理されないようにするため、ジョブ会計ジャーナル項目の処理に有用な 2 つのフィールド参照ファイルが提供されています。ファイル QSYS/QAJBACG4 にはレコード様式 QAWTJAJ4 が含まれ、JB 項目に使用されます。 ファイル QSYS/QAPTACG5 にはレコード様式 QSPJAPT5 が含まれ、DP または SP 項目に使用されます。 この同じ様式は、出力が SP (スプール) または DP (非スプール) かに関係なく、すべてのプリンター・ファイル項目に使用されます。 直接印刷ファイル用の DP 項目には使用されないフィールドも含まれ、こうしたフィールドにはブランクが含まれます。

以下は、使用可能な方法の一部です。

- 基本的な JB 項目および DP または SP 項目は、提供されているフィールド参照ファイル様式を使用して、なおかつ DSPJRN コマンドを JB に一度、さらに DP または SP に対して一度実行して、2 つの出力ファイルを作成すると処理できます。 このようにすると、2 つの物理ファイルに対して 1 つの論理ファイルを定義し、高水準言語プログラムを用いて外部記述ファイルを処理できます。
- 提供されているフィールド参照ファイル (QSYS/QAJBACG4) の 1 つを使用するファイルを作成すると、JB 項目のみを処理して外部記述ファイルを作成できます。その後、このファイルは QUERY ユーティリティーまたは高水準言語プログラムによって処理できます。
- QJORDJE のデフォルトの DSPJRN 様式を使用して、2 つのどちらのタイプのジャーナル項目も変換できます。 その後、プログラム記述ファイルを使用して、高水準言語プログラムでジャーナル項目を処理できます。

以下の DDS は、QSYS で QAJBACG4 フィールド参照ファイルを使用して、JB ジャーナル項目の物理ファイルを定義します。(物理ファイル作成 (CRTPF) コマンドを使用して) 同じ名前 (QAJBACG4) を持つファイルをモデル・ファイルとして作成できます。

```
R QAWTJAJ4 FORMAT(QSYS/QAJBACG4)
```

以下の DDS は、QSYS で QAPTACG5 フィールド参照ファイルを使用して、DP または SP ジャーナル項目の物理ファイルを定義します。 (CRTPF コマンドを使用して) 同じ名前 (QAPTACG5) を持つファイルをモデル・ファイルとして作成できます。

```
R QSPJAPT5 FORMAT(QSYS/QAPTACG5)
```

物理ファイルにもキー・フィールドを指定できますが、この例では、論理ファイルが順序付けに使用されています。同じ名前のメンバーがある 2 つの物理ファイル (JB 用に 1 つと、DP または SP 用に 1 つ) を作成する場合、以下の DSPJRN コマンドを発行してそれらの項目を変換できます。 ライブラリー YYYY に、同じ名前の物理ファイルをモデル・ファイルとして作成したと想定します。

```
DSPJRN JRN(QACGJRN) JRNCDE(A) ENTTYP(JB)
OUTPUT(*OUTFILE) OUTFILE(YYYY/QAJBACG4)
DSPJRN JRN(QACGJRN) JRNCDE(A) ENTTYP(SP DP)
OUTPUT(*OUTFILE) OUTFILE(YYYY/QAPTACG5)
```

DSPJRN コマンドの使用および選択基準を制御して、同じ項目を何度も変換できないようにします。 たとえば、特定の日付範囲のすべての項目を選択できます。ジョブ会計分析用に、すべての項目を特定の期限 (たとえば、月ごと) で変換できます。 1 つ以上のジャーナル・レシーバーが該当月の間に使用されました。 同じメンバーに対して DSPJRN コマンドを使用するごとに、新しい項目が追加される前にメンバーが消去されるということに注意してください。 DSPJRN コマンドの JOB パラメーターを使用しないでください。一部の項目はシステム・ジョブによるジョブに対して作成されるので、期待通りには出力されません。

両方の物理ファイルの処理を許可する:

以下の DDS を入力して、両方の物理ファイルの処理が可能な論理ファイルを作成します。これにより、会計コードの順序で単一ファイルを読み取り、高水準言語プログラムを使用して報告書を印刷できます。

```
R QAWTJAJ4  PFILE(YYYY/QAJBACG4)
K JACDE
R QSPJAPT5 PFILE(YYYY/QAPTACG5)
K JACDE
```

基本ジョブ会計レコードの処理

論理ファイルを使用して、基本ジョブ会計レコードのみをユーザー名ごとの会計コード順に処理したい場合、以下の DDS を論理ファイルに入力できます。

```
R QAWTJAJ4 PFILE(YYYY/QAJBACG4)
K JACDE
K JAUSER
```

この論理ファイルは、QUERY ユーティリティーまたは高水準言語プログラムによって処理できます。システムの異常終了が生じる場合、ジャーナル項目の JARES フィールドの最初の 30 バイトを使用する修飾ジョブ名には、リソースを使用するジョブではなく、次の IPL 時に項目を書き込むシステム・ジョブが記載されます。このため、 JB 項目で行われるすべての分析では JAJOB、JAUSER、およびJANBR フィールドを使用してください。

回復とジョブ会計

ジョブが異常終了すると、最終会計項目が書き出され、以前に書き出された会計項目すべてがジャーナルに出力されます。システムの異常終了が生じると、最終の経路指定ステップまたは最終の会計終了セグメント (どちらであれ最近に生じた) で以下の会計データが失われます。

- 印刷行数および印刷ページ数に関する情報
- 作成されたファイル数

- データベース書き込み、取得、および更新の各操作
- 通信読み取り操作および書き込み操作
- 補助入出力操作
- トランザクション時間
- トランザクション・フィールド数
- アクティブ状態時間
- 中断状態時間

システムの異常終了後、ジャーナルのジョブ完了時刻は CPF1164 メッセージ内のジョブ完了時刻と同じにはなりません。 このメッセージでは、システム終了に最も近い時間ではあるもののジョブ会計ジャーナル項目が IPL 時にジャーナルに送信される時間を使用し、そのジョブ完了時刻はシステムの異常終了が生じたときよりも後の、現行システム時間になります。

システムが異常終了すると、一部のジャーナル項目が失われる可能性があります。ジャーナルに書き込まれますが、ディスクに強制的に書き込まれない (これは、ジャーナル項目送信 (SNDJRNE) コマンドの FORCE(*NO) と同等) 項目は、失われる可能性があります。 以下が含まれます。

- 会計コード変更 (CHGACGCDE) コマンドによって生じる JB 項目
- DP および SP 項目

ジョブが完了すると、最終の会計コード項目がディスクに強制的に必ず書き込まれます (SNDJRNE コマンドで FORCE(*YES) が 指定される場合と同様)。 会計項目がディスクに強制的に書き込まれる場合、項目を生成したジョブにかかわらず、ジャーナル内に既に存在していた項目すべてがディスクに強制的に必ず書き込まれます。

例外

システムで *PRINT 会計のみが指定される場合、実行されるジョブ終了 FORCE(*YES) ジャーナル項目は存在しません。ですから、システムの異常終了時に失われないようにしたい重要な会計項目が CHGACGCDE コマンドによって書き込まれる場合には、SNDJRNE コマンドを発行して FORCE(*YES) オプションを指定できます。会計ジャーナルにファイルもジャーナルする場合、データベース変更はジャーナルに必ず強制的に書き込まれ、同時にそれにより既に存在していた会計項目もすべて強制的に書き込まれます。

システムの異常終了が生じる場合、またはジョブの独自の会計コード以外を変更する場合、ジャーナル項目の JARES フィールドの最初の 30 バイトを使用する修飾ジョブ名には、リソースを使用するジョブではなく、次の IPL 時に JB 項目を書き込むシステム・ジョブが記載されます。JAJOB、JAUSER、および JANBR フィールドは、分析目的で使用してください。

損傷したジョブ会計ジャーナルまたはジャーナル・レシーバー:

ジャーナルまたはその現行レシーバーで損傷が発生して会計項目をジャーナルできない場合、CPF1302 メッセージが QSYSOPR メッセージ待ち行列に送信され、会計データは CPF1303 メッセージ内の QHST ログに書き込まれます。ジョブはジャーナル項目を送信しようとしながら、通常通り実行されます。損傷したジャーナルまたはジャーナル・レシーバーの回復は、他のジャーナルの場合と同じです。

ジャーナル QACGJRN を、他のジョブによって割り振らないでください。ジャーナルを他のジョブによって割り振ると、ジャーナル項目はメッセージ・テキストに変更され、CPF1303 メッセージとして QHST ログに送信されます。

ジャーナル表示 (DSPJRN) コマンドで OUTFILE パラメーターを使用して、会計ジャーナル項目を処理可能なデータベース・ファイルに書き出すことができます。

さらに、QACGJRN ジャーナルでジャーナル項目受信 (RCVJRNE) コマンドを使用して、QACGJRN ジャーナルに書き込まれた項目として受信することもできます。 ジョブ会計ジャーナルまたはジャーナル・レシーバーが損傷すると、システムは操作を続行し、ヒストリー・ログに会計データを記録します。ジャーナルまたはジャーナル・レシーバーの損傷を回復するには、ジャーナル処理 (WRKJRN) コマンドを使用します。 ジャーナルまたはジャーナル・レシーバーの損傷が回復した後、ジャーナル会計情報 (QACGLVL) システム値を自分のインストールに適切な値に変更します。(QACGLVL システム値を変更しないと、システムは新しいジャーナル・レシーバーに会計情報を記録しません。)

CPF1303 メッセージへのアクセス:

CPF1303 メッセージの情報にアクセスするため、高水準言語プログラムを作成します。

CPF1303 メッセージと一致するレコードを定義するには、以下のフィールドを含めます。

System Time (システム時刻) Char (8)
Message Record Number (メッセージ・レコード番号) Bin (4)
Qualified Job Name (修飾ジョブ名) Char (26)
Entry Type (項目タイプ) (JB、DP、または SP) Char (2)
Length of Data (データ長) Bin (2)

以下がフィールドに続きます。

JB 項目用の JASPN による JAJOB
SP および DP 項目用の JABYTE による JAJOB

プログラム例については、「CL プログラミング」内のジョブ完了メッセージの QHST ファイルの処理について取り上げているセクションを参照してください。

CPF1164 メッセージは常に 3 つのレコードで構成され、CPF1303 メッセージは常に 4 つのレコードで構成されています。標準ジャーナル接頭部フィールドに含まれる情報は、このメッセージには含まれません。必要とされるのは、ジョブ終了、日付、および時間に関する情報です。この情報は、CPF1303 メッセージのレコード 1 にあります。

リファレンス

実行管理機能を使用しているときに、ここで取り上げる役に立つトピックを参照する必要が生じる場合があります。

(**「IBM i5/OS Information Center バージョン 6 リリース 1 (V6R1)」 → 「システム管理」 → 「実行管理機能」 → 「Reference (リファレンス)」**)

サーバー・ジョブ・テーブル
このサーバー・テーブルは、サーバー、サーバー・ジョブ、ジョブ記述、およびサブシステムがどのように相互にマップされているかを調べるための参照として使用することができます。

システム値ファインダー
システム値ファインダーは、システム値についての情報を検索するために使用します。 システム値のカテゴリーを、System i ナビゲーター に表示されるとおりに検索できます。または文字ベースのインターフェースで使用したシステム値の名前を検索できます。

実行管理機能 API

実行管理機能 API は、各種アプリケーションで使用される機能を実行します。実行管理機能 API ページでは、ジョブ、サブシステム記憶域プール、サブシステム・ジョブ待ち行列、データ域、ネットワーク属性、システム状況、システム値、およびフライト・レコーダーを取得および操作する API のリストが表示されます。実行管理機能出口プログラムのリストも含まれます。

IPL SRC ファインダー

IPL システム参照コード (SRC) ファインダーは、IPL の実行時にシステム上に表示される SRC メッセージについての情報を検出するために使用します。 SRC は IPL の状況を示し、多くの場合、問題分析に役立ちます。 SRC は名前で検索できますが、最も一般的な SRC のリストを表示することもできます。

グループ・ジョブ

以下のグループ・ジョブに関する情報は、以前の環境を保守する際の参照資料として組み込まれています。今日のコンピューティング環境では、1 つのワークステーションが各機能ごとにそれぞれセッションを持つのが一般的です。

グループ・ジョブは、システム要求キーを押すことによって要求される 2 次対話式ジョブと似ています。ただし、ワークステーションにサインオンするたびに最大 16 個 (2 次対話式ジョブがある場合は合計 32 個) のグループ・ジョブを開始することができ、アプリケーション・プログラムは中断をより簡単に処理することができます。

グループ・ジョブの利点

グループ・ジョブの利点のいくつかを以下にリストします。

- ワークステーション・ユーザーはアテンション・キーを押して、1 つの対話式グループ・ジョブの作業を中断し、他のいくつかの対話式グループ・ジョブのいずれかに切り替え、元のグループ・ジョブに戻るという作業を迅速に行うことができます。アテンション・キーはアテンション・プログラムのセット (SETATNPGM) コマンドによって有効にされ、グループ・ジョブに依存せずに使用することができます。
- グループ・ジョブをディスプレイ装置パススルーとともに使用することにより、ネットワーク内の多様なシステム上のさまざまな対話式ジョブでの切り替えを簡単かつ迅速に行うことができます。

グループ・ジョブの概念

- グループ・ジョブは対話式ジョブにのみ適用されます。
- 1 つのグループ内のグループ・ジョブの最大数は 16 個です (ユーザーが 2 次対話式ジョブに移動する場合はさらに 16 個使用できます)。
- グループ・ジョブはユーザーに対して固有です (複数のユーザーによって共有されません)。
- 1 度に 1 つのグループ・ジョブだけがアクティブになります (それ以外は中断状態になります)。
- グループ・ジョブは相互に依存せず、独自のジョブ・ログ、スプール・ファイル、ライブラリー QTEMP などを持ちます。
- グループ・ジョブは、グループ・ジョブへの移行 (TFRGRPJOB) コマンドによって呼び出されます。このコマンドは通常、ユーザー作成メニュー・プログラムから実行されます。このプログラムはアテンション・キーを押すことによって呼び出されます (その前に SETATNPGM コマンドを実行しておく必要があります)。

- 1 つのグループ・ジョブから別のグループ・ジョブにデータを渡すために、512 バイトのグループ・データ域を使用することができます。このグループ・データ域はグループ属性変更 (CHGGRPA) コマンドによって暗黙的に作成されます。グループ・データ域については「CL プログラミング」により詳しい情報が収められています。

グループ・ジョブの間の切り替え

非グループ・ジョブからグループ・ジョブに切り替えたり、グループ・ジョブから非グループ・ジョブ (それがグループ内で唯一のジョブである場合) に切り替えたりするには、グループ属性変更 (CHGGRPA) コマンドを使用します。

新規グループ・ジョブの作成

新規グループ・ジョブを作成するには、グループ・ジョブへの移行 (TFRGRPJOB) コマンドを使用します。

注: TFRGRPJOB コマンドを使用した後は毎回、SETATNPGM コマンドを使用してアテンション・キーをオンに設定する必要があります (必要な場合)。

1 つのグループ・ジョブから別のグループ・ジョブへの移動

同じグループ内の 1 つのグループ・ジョブから別のグループ・ジョブに移動するには、グループ・ジョブへの移行 (TFRGRPJOB) コマンドを使用します。

注:

1. TFRGRPJOB コマンドを使用した後は毎回、SETATNPGM コマンドを使用してアテンション・キーをオンに設定する必要があります (必要な場合)。
2. 更新操作を行っている場合は、別のグループ・ジョブに移動する前に、レコード・ロック検査 (CHKRCDLCK) コマンドを使用して、ジョブにレコード・ロックがないか検査します。

1 つのグループ・ジョブから別のグループ・ジョブへの制御の移動

アテンション・キー処理プログラムがある場合、1 つのグループ・ジョブから別のグループ・ジョブに制御を移動することができます。アテンション・キーが押されると、アテンション・キー処理プログラムは、(ユーザーがグループ・ジョブを選択する) メニューを表示するか、またはユーザーを別のグループ・ジョブに即時に移動させることができます。アテンション・キー処理サポートにより、1 つのジョブを終了して別のジョブに移動することなく、1 つのグループ・ジョブから別のグループ・ジョブに迅速かつ簡単に制御を移動することができます。

メニューを表示せずに別のグループ・ジョブに移動する

アテンション・キーを使用して、メニューを表示せずに直接別のジョブに移動することができます。例えば、グループ・ジョブ A のアテンション・キー処理プログラムがグループ・ジョブ B に移動することができ、さらにグループ・ジョブ B のアテンション・キー処理プログラムもグループ・ジョブ A に戻ることができるとします。この場合、1 度のキー・ストロークによって機能を切り替えることができます。

グループ・ジョブの終了

- グループ内の 1 つのグループ・ジョブを終了するには、グループ・ジョブ終了 (ENDGRPJOB) コマンドを使用します。
- グループ内のすべてのグループ・ジョブを終了するには、SIGNOFF コマンドを使用します。

注: ENDJOB コマンドはパラメーター ADLINTJOBS をサポートします。 *GRPJOB が指定され、JOB パラメーターに指定されたジョブがグループ・ジョブである場合、グループに関連付けられているすべてのジョブが終了します。

また、グループ・ジョブ終了 (ENDGRPJOB) コマンドはシグナル SIGTERM をサポートしていません。しかし、ジョブ終了 (ENDJOB) コマンドはシグナル SIGTERM をサポートしています。

グループ・ジョブの正常終了

環境によっては、ENDGRPJOB コマンドを発行するのではなく、エンド・ユーザーに特定のグループ・ジョブを正確に終了させることを強制する方が望ましい場合があります。例えば、複雑な更新が関係するグループ・ジョブをユーザーが持つ場合、ジョブを正常に終了させる必要があるかもしれません。別の例として、SEU セッション中にユーザーが機能を正常に完了する必要が生じる場合もあります。

これは、システムによって提供されるサポートによって達成できます。例えば、以下の指示に従うこともできます。

1. 各グループ・ジョブによってテストされるグループ・データ域のスイッチを、シャットダウン・スイッチとして機能するように設定します。つまり、スイッチをオンに設定すると、グループ・ジョブ機能は終了します。
2. RTVGRPA コマンドおよび GRPJOBL 戻り変数を使用することにより、アクティブなグループ・ジョブ名にアクセスします。
3. 正確に終了させる必要のあるグループ・ジョブ名の既定のリストに対して、アクセスされるそれぞれの名前 (2 番目のグループ・ジョブから始まる) を比較します。
4. グループ・ジョブ名がリストにない場合、ENDGRPJOB コマンドによって即時に終了させることができます。
5. ジョブを正確に終了させる必要がある場合は、TFRGRPJOB コマンドを使用してグループ・ジョブに移動します。

すべてのグループ・ジョブのアテンション・キー処理プログラムはシャットダウン・スイッチに敏感でなければならず、スイッチがオンに設定されている場合は別のグループ・ジョブに移動されません。

ユーザーがグループ・ジョブの機能 (例えば更新プログラム) を終了したときに発生することを制御する制御プログラムが各グループ・ジョブごとにある場合は、そのプログラムはシャットダウン・スイッチをテストし、戻りを実行することもできます。これによりグループ・ジョブが終了され、前のアクティブ・グループ・ジョブに制御が戻されます。

アテンション・キー処理プログラムは CHKRCDLCK コマンドを使用して、更新のためにロックされたレコードをアプリケーションが持つときに、ワークステーション・ユーザーがアテンション・キーを押したかどうか判別することができます。この場合、アテンション・プログラムはメッセージを送信し、アテンション・キーを使用する前に操作を完了するようユーザーに指示する場合があります。

グループ・ジョブの理論

CHGGRPA コマンドは現行ジョブをグループ・ジョブと見なし、グループ内で一意的に識別するためのグループ・ジョブ名を付けます (この時点でグループはグループ・ジョブを 1 つだけ持ちます)。各グループ・ジョブはユーザーに対して固有です。 2 人の異なるユーザーが同じグループ・ジョブを共用することはありません。ジョブがグループ・ジョブとして指定されると、新規グループ・ジョブを呼び出すことができるようになります。グループ・ジョブには制約事項もあります (RRTJOB、TFRJOB を使用できないなど)。グループ内に存在するアクティブ・ジョブが 1 つだけの場合、そのジョブは非グループ・ジョブになることができます。

グループ・ジョブの通信を可能にする

グループ・ジョブの相互通信を可能にするために、ジョブがグループ・ジョブとなるときに、グループ・データ域と呼ばれる 512 バイトの特別なデータ域が自動的に作成されます。グループ・データ域にアクセスできるのはグループ内のジョブだけです。データ域コマンドの DTAARA パラメーターに特殊値 *GDA を使用することによってアクセスします。

グループ・ジョブの呼び出し

グループ・ジョブを使用する際は、このセクションで説明されているようなアテンション・キー・メニューのアプローチは必要ありません。グループ・ジョブは、アプリケーション・プログラムか、または TFRGRPJOB コマンドの GRPJOB(*SELECT) パラメーターによって呼び出すことができます。

グループ・ジョブとシステム要求機能

グループ・ジョブ機能は、1 度に 1 つのジョブのみがアクティブになり、それ以外は中断状態になるという点で、システム要求機能と似ています。グループ・ジョブは、次の点でシステム要求と異なります。

- グループ・ジョブを開始するためにサインオンする必要はありません。同じユーザー・プロファイルおよび環境が使用されます。
- 1 度に最大 16 個のグループ・ジョブが存在できます。ユーザーが移動先のグループ・ジョブを選択する必要があるのに対し、システム要求ではユーザーは 2 つのジョブの間だけを移動します。通常、グループ・ジョブでは、アテンション・キーを押して表示されるメニューを使用して、ユーザーが移動先のグループ・ジョブを選択することができます。グループ・ジョブとシステム要求を併用して、1 人のユーザーが合計 32 個のグループ・ジョブを使用できるようにすることができます。ただし、これら 32 個のジョブは 2 つのグループに分かれており、各グループは独自のグループ・データ域および他のグループ属性を持ちます。
- システム要求機能では、キーボードがロックされ、アプリケーションの機能が進行している間に、ワークステーション・ユーザーがジョブを中断することができます。これによってイベントの論理順序が中断される場合があります。例えば、レコードがロックされたままになる可能性があります。一方、アテンション・キーはキーボードが入力のためにアンロックされる場合にのみアクティブになります。また、アプリケーションはアテンション・キーがアクティブになるタイミングを制御することができ、不適切なときにそれが使用されないようにすることができます。システム要求機能は、ワークステーション・ユーザーに権限があればいつでも使用することができます。

  注: 事前システム要求プログラム出口プログラムは、ユーザーがシステム要求キーを押したときに呼び出されます。ユーザーがシステム要求キーを押すと、オペレーティング・システムは登録機能を介してユーザー作成の出口プログラムを呼び出します。入出力に 1 つのパラメーターが使用されます。登録機能から出口プログラムが呼び出された後、「システム要求」メニューがその表示フラグで戻される値に基づいて呼び出されます。追加情報については、「System API Reference」を参照してください。

アテンション・キー処理プログラム

特定の呼び出しレベルのプログラムをアテンション・キー処理プログラムとして特定することができます。アテンション・キー処理プログラムは、SETATNPGM コマンドを発行したプログラムと同じジョブで実行され、同じジョブ属性、オーバーライド、およびグループ権限を持ちます。しかし、プログラムによって採用される権限は、中断されたプログラムからは発生しません。アテンション・キー処理プログラムをユーザー・プロファイルで指定することもできます。

プログラムをアテンション・キー処理プログラムとして特定する

プログラムをアテンション・キー処理プログラムとして特定するには、SET(*ON) を指定してアテンション・プログラムのセット (SETATNPGM) コマンドを使用します。このコマンドは、コマンドを実行しているジョブのその呼び出しレベルでこのプログラムを特定します。アテンション・キーが押されると、実行中のジョブが停止され、表示が保存されて、アテンション・キー処理プログラムが呼び出されます。アテンション・キー処理プログラムを呼び出すときに、パラメーターは渡されません。

注: ユーザーがシステム・アテンション・キーを押すと、事前アテンション・プログラム出口プログラムが呼び出されます。ユーザーがシステム・アテンション・キーを押すと、オペレーティング・システムは登録機能を介してユーザー作成の出口プログラムを呼び出します。入出力パラメーターはありません。登録機能から出口プログラムが呼び出された後、システム・アテンション・プログラムが呼び出されます。

アテンション・キーの状況に対する呼び出しレベルの影響

SETATNPGM コマンドは、呼び出し指向コマンドです。つまり、ある特定の呼び出しレベルで SETATNPGM コマンドが発行されると、現行およびそれより下位の呼び出しレベルのアテンション・キー処理プログラムが有効になります。これは、次の SETATNPGM コマンドが実行されてアテンション・キー処理プログラムまたはアテンション・キーの状況が変更されるまで続きます。 SETATNPGM プログラムを発行したプログラムが戻るときは常に、表示が復元され、アテンション・キー処理プログラムおよびアテンション・キーの状況が現行の呼び出しの前の状態にリセットされます。 RETURN コマンドの代わりに制御権移動 (TRFCTL) コマンドを使用すると、移動されたプログラムが戻るまで状況はリセットされません。

アテンション・キーを使用するべき時

アテンション・キーを使用して、アテンション・キー処理プログラムを呼び出します。通常のワークステーションの使用では、アテンション・キーはキーボードがアンロックされている場合、つまり、プログラムの入力準備が整っている場合にのみ押すことができます。これは、読み込みまたは書き込み/読み込み操作が発行されるか、または書き込み操作で UNLOCK DDS キーワードが使用される場合に発生します。

アテンション・キーの使用は、アプリケーション・プログラムが中断のタイミングを制御するという点で、システム要求キーの使用と異なります。

例外

アプリケーション・プログラムが複数の装置ファイルに対して即時ゲット操作を実行する場合はこの例外が発生します。アテンション・キーを押すと、それらのプログラムはどの時点であったとしてもアテンション・キー処理プログラムによって中断されます (入力禁止信号がオンになっている場合でも、キーボードは即時ゲット操作時にアンロックされます)。そのため、依存機能を実行するアプリケーション・プログラムは (特に即時ゲット操作時には)、依存コードの前に SETATNPGM PGM(*CURRENT) SET(*OFF)、後に SETATNPGM PGM(*CURRENT) SET(*ON) を実行することにより、保護する必要があります。

注: 高水準言語プログラムは、QCMDEXC を呼び出すことによって SETATNPGM コマンドを使用することができます。

アテンション・キーを使用するべきでない時

次のような条件が存在する場合は、アテンション・キーを使用してアテンション・キー処理プログラムを呼び出すことができません。

- キーボードがロックされている (即時ゲット操作に関する前述の例外にご注意ください)。

- システム要求メニューまたはそのいずれかのオプションが使用されている。
- メッセージの表示画面が表示されている。
- i5/OS ライセンス・プログラムがすでにアテンション・キー処理プログラムを呼び出し、すでにアクティブにしている (ただし、プログラムがもう一度 SETATNPGM を発行すると、アテンション・キーは使用可能になります)。
- BASIC セッションが進行中であるか、BASIC プログラムが呼び出されている。

アテンション・キーと BASIC セッション

BASIC セッションでは、アテンション・キーが必要に応じて BASIC によって処理されます。例えば、SETATNPGM コマンドによってアテンション・キーがオンに設定された後 BASIC プログラムが呼び出された場合、アテンション・キーは BASIC によって処理されます。 BASIC プログラムの終了後、アテンション・キー処理プログラムは再び有効になります。

アテンション・キー処理プログラムのコーディングのヒント

アテンション・キー処理プログラムを定義する際には注意が必要です。アテンション・キーが押されると、アテンション・キー処理プログラムは進行中のプログラムと同じジョブで実行されるからです。そのため、中断されたプログラムは、保持しているロックによって保護されません。中断されたプログラムがオブジェクトに対する排他ロックを持っている場合、アテンション・キー・プログラムは、同じジョブで実行されるので、排他ロックを持つジョブの一部となります。

アテンション・キー処理プログラムを定義する際は、次のガイドラインに従うことをお勧めします。

- 単純な機能を使用します (例えば、ワークステーション・ユーザーが別のグループ・ジョブまたは 2 次対話式ジョブに移動するためのメニュー)。
- アテンション・キーを押すときは、使用中になっている可能性のあるオブジェクトまたは機能の参照を避けます。
- アテンション・キーを押すときは、非再帰的関数の呼び出しを避けます。非再帰的関数とは、中断した後再び呼び出すことのできない関数のことです。関数の多くは非再帰的です (例えば、DFU のような高水準言語プログラムおよびユーティリティー)。
- ワークステーション・ユーザーがコマンド入力画面を現行ジョブの一部として表示するためのオプションを指定することは避けます。ユーザーがプログラマーである場合には、コマンド入力画面のオプションを含むメニューを表示することは意味のあることです。コマンド入力画面は個々のグループ・ジョブとして (例えば、TFRGRPJOB コマンドで INLGRPPGM(QCMD) を指定することによって) 指定する必要があります。これによって、すでに使用中のオブジェクトが重複して使用されることのないようにします。
- アテンション・キー処理プログラムは、アテンション・キーが押される前に進行中だったプログラムによって採用されている権限を持ちません。
- アテンション・キー処理プログラムは独自のデータ域 (*LDA) を持ちません。 1 つのジョブにつきローカル・データ域は 1 つのみであり、アテンション・キー処理プログラムは中断されるプログラムと同じジョブで実行されるため、両方のプログラムが同じローカル・データ域を共有します。
- 送信勧誘装置からの読み取り操作は、アテンション・キー処理プログラムの実行中にタイムアウトになる可能性があることを知っておく必要があります。そのため、アテンション・キー処理プログラムの実行中に進行中のプログラムのタイムアウトが完了する場合、そのタイムアウトの結果として取られるアクションはすべて、進行中のプログラムに戻るときに発生します。例えば、次の条件が満たされている場合、プログラムはアテンション・キー・ハンドラーから戻るときに終了します。
  - ファイルの WAITRCD 値が 60 秒に設定されている。

– 1 分間キーが押されなかった場合にプログラムが終了するように設定されている。
– アテンション・キー・プログラムが呼び出され、1 分より長く実行される。

しかし、タイムアウトの完了を検査する前に使用可能なデータの検査が行われるため、注意が必要です。アテンション・キー・ハンドラーが離れた直後にキーが押されると、送信勧誘装置からの読み取りを完了するデータが使用可能になり、タイムアウトは検査されません。これにより、予期しない結果になる可能性があります。

グループ・ジョブのパフォーマンスに関するヒント

このトピックでは、グループ・ジョブの使用時に優れたシステム・パフォーマンスを保つためのいくつかのヒントを示します。

- システムに中断されたジョブが多数あったとしても、専用主記憶域要件が要因でなければ、通常その影響はわずかです。
- TFRGRPJOB コマンドが実行され、新規ジョブを開始する必要がある場合、発生するオーバーヘッドはシステムへのサインオンとほぼ同じです。コマンドが実行され、グループ・ジョブがすでに開始されている場合、必要となるオーバーヘッドは、2 次ジョブがすでにアクティブになっているときに「システム要求」メニューにある 2 次ジョブ・オプションへの転送を使用する場合とほぼ同じです。
- グループ・ジョブが任意の頻度で実行される場合は、それを終了させないのが望ましいでしょう。つまり、プログラムは終了せず、TFRGRPJOB コマンドを発行して、グループ・ジョブの機能が必要になるたびにジョブを開始しないということです。
- SETATNPGM コマンドを実行すると、アテンション・キーが押されたときに現行の表示が保存され、アテンション・キー処理プログラムが終了したときにそれが復元されます。これは「システム要求」メニューの使用とほぼ同じです。これには、リモート・ワークステーションに対してより顕著な影響があります。
- システムでアクティブなジョブの数の制御 (CRTSBSD コマンドの MAXJOBS パラメーター) は、任意の時点でアクティブなグループ・ジョブの数に影響を受けません。
- ジョブ構造の作成を制御するシステム値 (QACTJOB と QADLACTJ、および QTOTJOB と QADLTOTJ) はすべて影響を受けます。グループ・ジョブの追加を許可する場合、これらの値を増やす必要が生じる場合があります。

実行管理機能のトラブルシューティング

このトピックは、実行管理機能で発生する最も一般的な問題をトラブルシューティングする際に役立ちます。

ジョブのハングアップ

この表は、ジョブがハングアップする場合の考えられる理由をリストしています。

ジョブがオブジェクトのロックを待っている		
	診断方法	System i ナビゲーターでジョブの状況を参照します。『ジョブの状況の判断』を参照してください。ロックを待っているジョブの状況は、ロックの待機中 (*Waiting for lock*) です。
	回復	そのジョブに関するロック・オブジェクトのリストを参照して、ジョブがロックを待機しているオブジェクトを判別します。そのオブジェクトに対してホルダー・ロック・アクションを使用して、すでにロックを保持しているジョブを判別します。そして、そのジョブがロックを保持している理由と、そのロックを解放するためには何を行えばよいかを判別します。

ジョブが保留中		
	診断方法	System i ナビゲーターでジョブの状況を参照します。『ジョブの状況の判断』を参照してください。
	回復	ジョブを右マウス・ボタンでクリックして、「**解放**」をクリックします。

ジョブ待ち行列でジョブがハングアップする場合の理由としては、以下の事柄が考えられます。

ジョブ待ち行列が保留中		
	診断方法	System i ナビゲーターで、ジョブ待ち行列の状況を参照します。
	回復	1. 保留されていないジョブ待ち行列にジョブを移動します。『他のジョブ待ち行列へのジョブの移動』を参照してください。 2. ジョブ待ち行列を解放します。そのために、ジョブを右マウス・ボタンでクリックして、「**解放**」をクリックします。

アクティブ・サブシステムによってジョブ待ち行列が割り振られていない		
	診断方法	System i ナビゲーターで、ジョブ待ち行列の状況を参照します。
	回復	1. アクティブ・サブシステムによって割り振られているジョブ待ち行列にジョブを移動します。『他のジョブ待ち行列へのジョブの移動』を参照してください。 2. このジョブ待ち行列に関するジョブ待ち行列項目が入っているサブシステムを開始します。『サブシステムの開始』を参照してください。 3. ジョブ待ち行列項目の追加 (ADDJOBQE) コマンドを使用して、このジョブ待ち行列に関するジョブ待ち行列項目をアクティブ・サブシステムに追加します。

サブシステムの最大数に達した		
	診断方法	System i ナビゲーター で、サブシステムの最大アクティブ・ジョブの値を参照します。そのために、サブシステムを右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。
	回復	1. ジョブを別のジョブ待ち行列に移動します。『他のジョブ待ち行列へのジョブの移動』を参照してください。 2. 最大値を増やします。そのためには、「サブシステム記述変更」(CHGSBSD) コマンドを使用します。

ジョブ待ち行列の最大値に達した		
	診断方法	System i ナビゲーター で、ジョブ待ち行列の「最大アクティブ・ジョブ」の値を参照します。そのためには、ジョブ待ち行列を右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。その後、「**アクティビティー**」タブを選択します。
	回復	1. ジョブを別のジョブ待ち行列に移動します。『他のジョブ待ち行列へのジョブの移動』を参照してください。 2. 最大値を増やします。そのために、「ジョブ待ち行列項目変更」(CHGJOBQE) コマンドを使用します。

優先順位レベルの最大値に達した		
	診断方法	プロパティーを調べることによって、そのジョブのジョブ待ち行列優先順位を判別します。 System i ナビゲーター で、ジョブ待ち行列のジョブ優先順位値ごとの「最大アクティブ・ジョブ」を参照します。 そのためには、ジョブ待ち行列を右マウス・ボタンでクリックして、「**プロパティー**」をクリックします。その後、「アクティビティー」タブを選択して、「**拡張**」ボタンをクリックします。
	回復	1. ジョブを別のジョブ待ち行列に移動します。『他のジョブ待ち行列へのジョブの移動』を参照してください。 2. ジョブのジョブ待ち行列優先順位を変更します。『ジョブ待ち行列の優先順位の指定』を参照してください。 3. 最大値を増やします。そのために、「ジョブ待ち行列項目変更」(CHGJOBQE) コマンドを使用します。

ジョブのパフォーマンスが悪い

ここでは、ジョブのパフォーマンスが低下する理由に関する情報が取り上げられています。

メモリーが足りない		
	診断方法	ジョブのプロパティーを表示して、ジョブがどのメモリー・プールで実行されているかを判別します。その後、System i ナビゲーター でメモリー・プールのプロパティーを表示します。『メモリー・プールの使用状況の検査』を参照してください。 プールで障害が起きる比率が高い場合、プールのメモリーが十分ではないか、プールにあるジョブが多すぎてメモリーを競合していることを示しています。
	回復	1. システム・チューナーを使用していない場合は、オンにします。メモリー・プールとアクティビティー・レベルの自動調整について詳しくは、『パフォーマンス・システム値: メモリー・プールとアクティビティー・レベルの自動調整』を参照してください。 2. 可能であれば、処理しているプール内のメモリーの量を増やすか、またはメモリー・プールのアクティビティー・レベルを減らすかして、手動でそのプールを調整します。さらにマシン・プールを検査して、システム上のすべてのジョブが、使用中のメモリー・サイズに影響されていないことを確認することもできます。

アクティビティー・レベルが低すぎる		
	診断方法	ジョブのプロパティーを表示して、ジョブの状況およびどのメモリー・プールでジョブが実行されているかを判別します。ジョブの状況がアクティビティー・レベルを待機中 (*Waiting for activity level*) であった場合は、System i ナビゲーター でメモリー・プールのプロパティーを表示します。『メモリー・プールの使用状況の検査』を参照してください。 プールで不適格状態に推移する比率が高い場合、プールにあるジョブが多すぎてメモリーを競合していることを示しています。
	回復	1. システム・チューナーを使用していない場合は、オンにします。メモリー・プールとアクティビティー・レベルの自動調整について詳しくは、『パフォーマンス・システム値: メモリー・プールとアクティビティー・レベルの自動調整』を参照してください。 2. メモリー・プールのアクティビティー・レベルを増やして、手動でプールを調整します。

CPU リソースが不足している		
	診断方法	System i ナビゲーター の「アクティブ・ジョブ」リストでジョブの「CPU %」欄を確認します。 システムの使用率が高い場合、ユーザーのジョブの処理を完了させるだけの十分な CPU リソースを得ることができない可能性があります。
	回復	1. 可能であれば、システムの不要な作業を終了させるかまたは保留にします。 2. 少数のジョブが CPU を占有しているのであれば、それらのジョブの実行優先順位を変更します (実行優先順位の値が大きいということは、そのジョブの実行優先順位が低いということです)。

メモリー・プール・ページング・オプション		
	診断方法	アプリケーションがディスク処理の多いものであり、CPU 稼働率が低く、メモリーが十分にある場合は、エキスパート・キャッシュを使用すると効果的です。
	回復	エキスパート・キャッシュは、 System i ナビゲーター で共用メモリー・プールの「ページング」オプションを「計算済み (Calculated)」に変更して、オンにすることができます。 「ページング」オプションは、メモリー・プールの「**プロパティー**」ページの「**構成**」タブにあり、共用プール (私用プールではなく) でのみ使用可能です。

ジョブの実行優先順位が低い		
	診断方法	システム上の他のジョブと比較してジョブの実行優先順位を判別する場合は、『ジョブ属性の表示』を参照してください。
	回復	他のジョブと比べて実行優先順位が低く (数字が大きい)、かつ優先順位の高い (数字が小さい) ジョブが CPU リソースのほとんどを使用しているために使用できる CPU があまり残されていないという場合、ジョブの実行優先順位を高くする必要があるかもしれません。『ジョブ属性の表示』を参照してください。さらに、システムの CPU 使用率が高く、ジョブの実行優先順位が低い場合は、『パフォーマンス・システム値: 優先順位バンド内でのジョブ優先順位の動的調整』および『パフォーマンス・システム値: 対話型ジョブのジョブ優先順位の動的調整』を参照してください。システム値を設定すると便利です。

パフォーマンスの詳細については、『パフォーマンス』を参照してください。システムのパフォーマンスを調整する方法の詳細については、『パフォーマンスのチューニング』を参照してください。

事前開始ジョブの調査

このトピックでは、「どのように事前開始ジョブの実ユーザーを検出し、事前開始ジョブが使用するリソースを判別するか」という問いに答えるために役立ちます。

System i ナビゲーター

System i ナビゲーターの実行管理機能およびマネージメント・セントラルを使用して、システム上で生じていることのリアルタイム分析を、それぞれ表示またはモニターすることができます。

1. アクティブ・サーバー・ジョブおよび現行ユーザーを表示するには、サーバー・ジョブ・ビューを使用します。 (「**ユーザー接続**」 → 「**接続 (connection)**」 → 「**実行管理機能**」 → 「**サーバー・ジョブ**」)
   - サーバー・ジョブ・コンテナーを右マウス・ボタン・クリックし、「**この表示をカスタマイズ (Customize this view)**」 → 「**列 (Columns)**」を選択して、現行のユーザー、合計 CPU 時間、および合計 CPU DB 時間が「**表示する列 (Columns to be displayed)**」リストにあることを確認します。

- アクティブ・サーバーのジョブ・リストが長大な場合、表示内容をジョブ名、ジョブ番号、現行ユーザー、または状況によって制限できます。サーバー・ジョブ・コンテナーを右マウス・ボタン・クリックして、「**この表示をカスタマイズ (Customize this view)**」 → 「**組み込み**」の順にクリックします。
- アクティブ・サーバーのジョブ・リストの表示順序は、列見出しをクリックしてソートできます。
- 画面は定期間隔で自動的に最新表示するように設定できます。 (「**ユーザー接続**」 → **サーバーを右マウス・ボタン・クリック** → 「**この表示をカスタマイズ (Customize this view)**」 → 「**自動最新表示 (Auto Refresh)**)。これにより即時変更を発見できます。

関心対象のジョブを見つけたら、ジョブを右マウス・ボタンでクリックして、ジョブの呼び出しスタック、ジョブ・ログ、経過パフォーマンス統計、最終 SQL ステートメント、およびジョブのプロパティーにアクセスできます。

2. 全 CPU 使用率をモニターするシステム・モニターをセットアップするには、マネージメント・セントラルを使用します。 (「**マネージメント・セントラル**」 → 「**モニター**」を展開し、「**システム**」を右クリックして、「**新しいモニター**」を選択します。)
   - モニターの実行中に、ポイントの 1 つをクリックして次のレベルの詳細を表示することができます。たとえば、CPU 使用率をモニターする場合、最も高い CPU 使用率を持つジョブのリストを表示できます。次いで CPU を多量に使用しているジョブを右マウス・ボタンでクリックし、「**プロパティー**」をクリックして、ジョブのプロパティーを表示できます。 (システム・モニターの使用方法について詳しくは、オンライン・ヘルプを参照してください。)
3. 適切なサーバー・ジョブをモニターするようにジョブ・モニターをセットアップし、それらのジョブがリソースを過剰に消費し始めた場合に通知を出すには、マネージメント・セントラルを使用します。 (「**マネージメント・セントラル**」 → 「**モニター**」を展開し、「**ジョブ**」を右クリックして、「**新しいモニター**」を選択します。)

文字ベース・インターフェース

コマンド: アクティブ・ジョブ処理 (WRKACTJOB)

このコマンドは、初期スレッド (ジョブが単一スレッドの場合のジョブ) の現行ユーザーを表示します。これは GUI に表示されるものと同じデータです。

関連概念

54 ページの『事前開始ジョブ項目』
事前開始ジョブ項目を使用して、事前開始ジョブを定義します。事前開始ジョブ項目は、装置割り振りまたはプログラム開始要求割り当てには影響を与えません。

16 ページの『サーバーの事前開始ジョブ』
事前開始ジョブ・モデルには、通常デーモン・ジョブまたは受話者ジョブと呼ばれる基本聴取ジョブが 1 つと、クライアント要求を処理する複数のサーバー・ジョブが存在します。デーモン・ジョブは、接続要求用のポートで listen します。新しい接続を受け取ると、デーモンは一般処理を幾らか行い、待機中の事前開始サーバー・ジョブにソケット記述子を渡します。

関連タスク

184 ページの『事前開始ジョブ項目の追加』
事前開始ジョブ項目は、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの入力時に開始できる事前開始ジョブを示します。 事前開始ジョブ項目は、文字ベース・インターフェースを使用してサブシステム記述に追加できます。

189 ページの『事前開始ジョブ項目の変更』
指定されたサブシステム記述内の事前開始ジョブ項目は変更可能です。事前開始ジョブ項目の変更時

に、サブシステムはアクティブであってもかまいません。サブシステムがアクティブであるときに項目に加えられた変更は、時間を経て反映されます。コマンドの発行後に開始されたすべての新規事前開始ジョブは、新規のジョブ関連値を使用します。このコマンドは、サブシステムの開始時または事前開始ジョブ開始 (STRPJ) コマンドの発行時に開始できる事前開始ジョブを識別します。

193 ページの『事前開始ジョブ項目の除去』
文字ベース・インターフェースを使用して、事前開始ジョブ項目をサブシステム記述から除去できます。現在アクティブ状態のいずれかのジョブが事前開始ジョブ項目を使用して開始された場合には、その項目を除去することはできません。

実行管理機能の関連情報

Information Center の他の一連のトピックには、実行管理機能に関する一連のトピックに関連した情報が含まれています。

体験レポート

実行管理機能の体験レポートは、日常のタスクに実行管理機能ツールを使用するための、実際的で現実的な方法を示しています。

ネットワーキング

ネットワーキング・テクノロジーを理解しておくことは、会社の e-business ソリューション全体において非常に重要です。ビジネスをインターネットと関連付け、E メールを構成し、マルチメディア・オブジェクトのサービスを Web ブラウザー・クライアントに提供する方法について確認してください。ファイルおよび印刷サービス、ユーザー・プロファイル管理、およびネットワーク操作を統合できます。サーバーに統合できる Windows サーバーについての情報を探し、リソースの保護に役立つセキュリティー・オファリングについてお読みください。

ネットワーク属性取得 (QWCRNETA) API

ネットワーク属性取得 (QWCRNETA) API を使用すると、ネットワーク属性を取得できます。

IPL 属性の取得 (QWCRIPLA) API

IPL 属性の取得 (QWCRIPLA) API は、IPL の実行中に使用される属性の設定を戻します。この API は、IPL 属性の表示 (DSPIPLA) コマンドと類似したサポートを提供します。

パフォーマンス

システム・パフォーマンスに影響を与えるさまざまなプロセスすべてを理解することは、経験の浅いユーザーにとっては挑戦となります。パフォーマンス上の問題を解決するには、要件やサポートされる機能がそれぞれ異なる多くのツールを効果的に使用する必要があります。パフォーマンス・データを収集して分析した後でも、その情報をどう処理するかを理解することは大きな課題となります。このトピックでは、パフォーマンス管理に関連したタスクおよびツールの一連の手引きを記載しています。

Performance explorer

Performance Explorer は、特定のアプリケーション、プログラムまたはシステム・リソースに関するより詳細な情報を収集し、特定のパフォーマンス上の問題を詳しく洞察します。これには、複数のタイプおよびレベルのトレースを実行する機能と、明細報告書を実行する機能が含まれます。

時間管理

System i ナビゲーター の時間管理コンポーネント内で、タイム・ゾーンおよび時刻調整機能を処理できます。 これらの機能を使用して、システムがシステム時刻を使用および調整するためのタイム・ゾーンを選択することができます。

システム値

システム値は、システムの操作環境に影響を与える情報です。システム値は、システム上のオブジェクトではありません。システム値には、システムの特定の部分の操作のための制御情報が含まれます。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、IBM 機械コードのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

本書「Work management」には、プログラムを作成するユーザーが IBM i5/OS のサービスを使用するためのプログラミング・インターフェースが記述されています。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

以下は、IBM Corporation の商標です。

DB2
Domino
i5/OS
IBM
IBM(logo)
IPDS
Lotus Notes
OS/400
SP
System i
System i/36
WebSphere

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、 IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、第三者の権利の不侵害の保証、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12