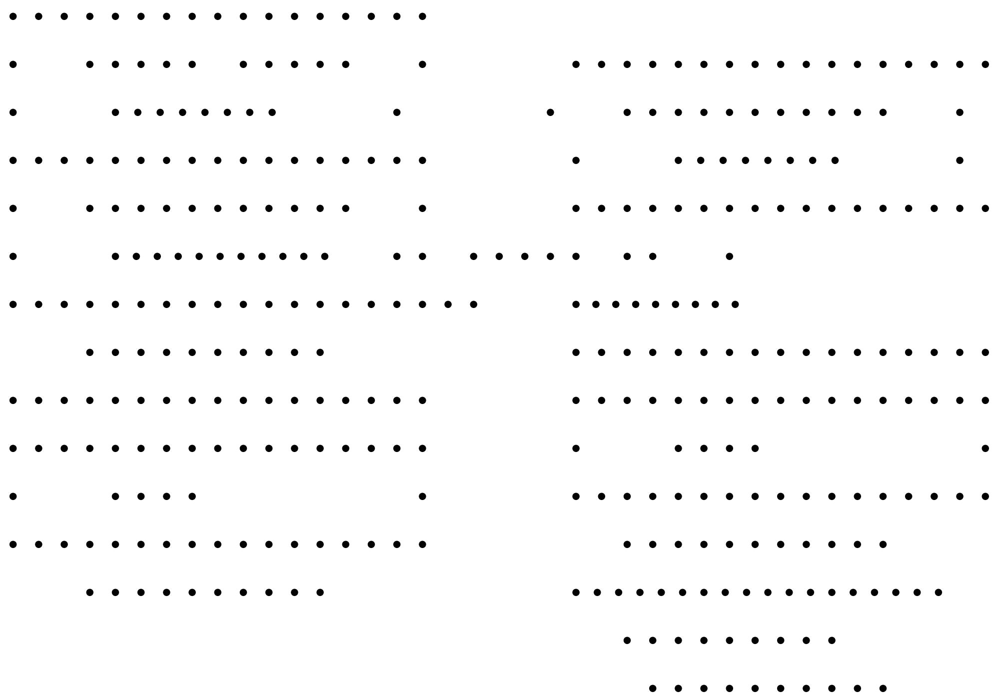
Outline

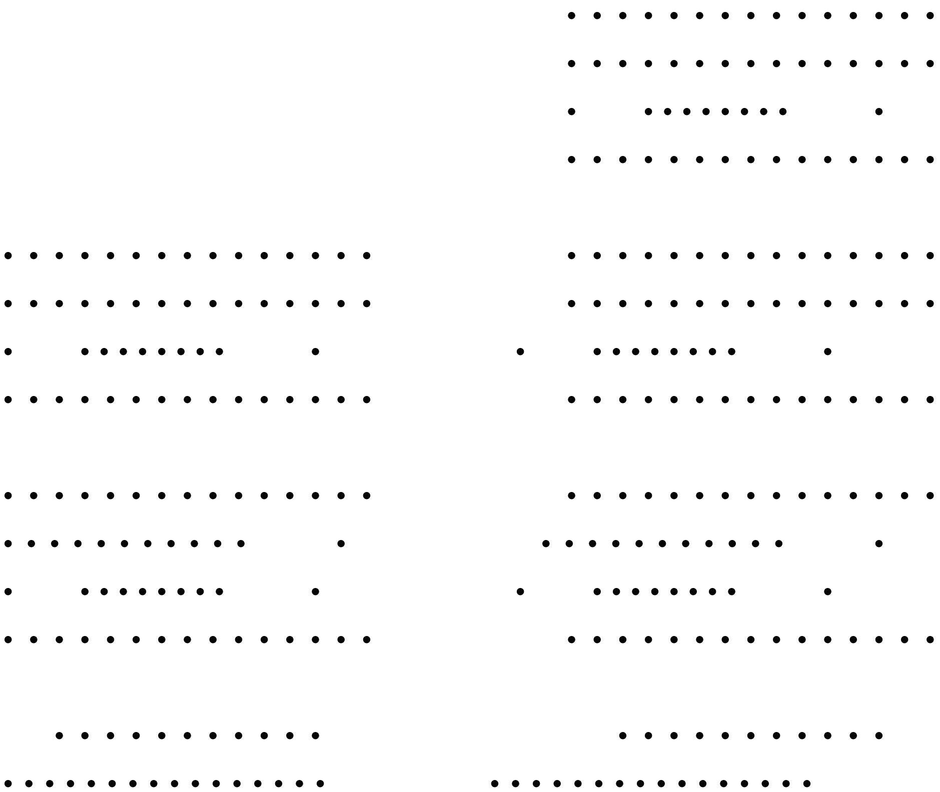
- Product overview
- News update from IBM COBOL development
- Migration overview
- Planning for Migration
- Benefits of Migration
- Preparation for Migration
- Using the new run time
- Testing your applications
- Moving LE into production
- COBOL source conversion
- Year 2000
- Technical Advantages of LE/370

Obsolete/Stabilized products



Strategic Products

Obsolete/Stabilized run-time libraries



Strategic Run-time libraries

Language Environment - complete picture

All of the programs supported by LE R5

News update

- Available in October, 1995:
  - A new release of COBOL/370:
    - IBM COBOL for MVS & VM Release 2 (5688-197)
  - IBM COBOL for workstations:
    - IBM VisualAge for COBOL for OS/2 (5622-793)
    - IBM COBOL Set for AIX (5765-548)

- Available in September, 1995:
  - A new release of LE/370:
    - IBM Language Environment for MVS & VM Release 5 (5688-198)

- Available in March 1995:
  - A replacement for VS COBOL II and DOS/VS COBOL:
    - COBOL for VSE and LE for VSE
    - PL/I too!

- Announced in October 1994:
  - A tool to help you migrate!
    - EDGE Portfolio Analyzer tool (5633-009)
    - Ask IBM to use option 3 ″Software Ordering Information″ in the HONE system

- VSAM control blocks above the line
  - VSAM buffers already above for XA/ESA systems
  - VS COBOL II Release 4 - APAR PN72097
  - LE/370 Release 3 and 4 - APAR PN72983
  - In base LE Release 5 product
  - Solved a unique(?) customer problem, helps everyone
- COBPACK serviceability APAR for VS COBOL II R4 and LE/370
  - VS COBOL II Release 4 - APAR PN59231
  - In base LE/370 Release 4 product
  - Many customers APPLY maintenance without ACCEPTing it
  - Before this new procedure, rebuilding COBPACK used modules from distribution libraries only
  - Allows rebuilding of COBPACKs without losing temp fixes by using modules from target libraries

- COBOL for MVS & VM Release 2 programs will NOT run on LE R4
  - LE for MVS & VM Release 4 books say they will
  - You must migrate to LE for MVS & VM Release 5 before you can use the COBOL for MVS & VM Release 2 compiler
- COBOL for MVS & VM Release 2 has 15 new reserved words:
  - CLASS-ID END-INVOKE INHERITS INVOKE
  - LOCAL-STORAGE METACLASS METHOD
  - METHOD-ID OBJECT OVERRIDE RECURSIVE
  - REPOSITORY RETURNING SELF SUPER
- APAR PN71388 for COBOL/370 provides an alternate reserved word table for Release 1 that will flag Release 2 reserved words with an informational message.
- COBOL for MVS & VM Release 2 will provide an alternate reserved word table for users who don't need the new function, and want compatibility with VS COBOL II reserved words.

Migration overview

- Plan, plan, plan
- Communicate with management and users
- Migrate runtime first, then upgrade source
- Use old compiler until new run time in production
- Migrate in test environment first
- Regression test...why?
  - Your code
  - Our code
  - Their code
- Migrate tested applications to production environment

Everything you need to know!

- Migration publications
  - IBM COBOL Compiler and Run-Time Migration Guide (GC26-4764)
  - IBM PL/I Compiler and Run-Time Migration Guide (SC26-3118)
  - IBM C/C++ for MVS/ESA Compiler and Run-Time Migration (SC09-2002)
  - IBM Language Environment for OS/390 & VM Run-Time Migration (SC26-8232)
  - Call (800)879-2755 to order IBM publications
- Other resources
  - COBOL CFORUM (Available though TALKLINK of IBMLINK)
  - http://www.software.ibm.com/ad/cobol/cobol.htm

Migration overview

- Rex Widmer′s fundamental rules for migration

  1. The run-time package must be at a product and maintenance level equal to, or later than, any compiled code which is executed under the run-time package.

  2. All dynamic run-time package components must be at a product and maintenance level equal to, or later than linked-in run-time components which are executed under the run-time package.

- Your migration plan may have long term effects. Example:

  - If you have all NORES now and you install LE/370 in LINKLST/LPA

  - Every relink will mean a newly migrated application

  - NORES behavior to RES behavior without recompile

  - Migration could still be going on in 10 years!

- I recommend planning to get it all done with the same technical staff available for problems within as short a period as possible

Planning for migration

- Are you ready for LE/370?
  - MVS/ESA V3.1.3 or later (or VM/ESA)
  - If CICS, then must be CICS V3.3 or later
    - no CICS required if you don′t use it!
  - NO macro-level CICS applications
    - could take a while to convert, don′t need to buy LE/370 until you have converted to Command-level
  - Other products may need to be updated
    - TCP/IP 3.1, BookManager 1.3, OfficeVision/MVS 1.3, Netview 3.1
  - Run-time migration, source conversion
- Are your vendors ready?
  - Vendor Interface document on MKTTOOLS
    - (LEVNDRST PACKAGE)
  - Help us out, call your vendors! We have...
  - If your vendor has not heard the word
    - Ask them to fax their request for info to (914)435-1752
    - Fax us the company name, product name, and phone number

Planning for migration

1. Develop a cost/benefit analysis
   - Include YEAR 2000, client-server, OO in your benefit
   - See ″Technical Advantages of LE/370″
2. Take inventory of your COBOL development tools
3. Assess requirements of new software
   - Prerequisite product levels
   - Compatibility with vendor products
   - Compatibility with IBM products
4. Take inventory of existing applications
5. Assess migration effort for each application
6. Schedule training for programmers

Migration and Year 2000 conversion

- Share the same process, and most of the same steps
  - Planning
  - Inventory
  - Ownership
  - Status
  - (Source conversion/Year2000 changes)
  - Compile, link
  - Unit Test
  - System Test
- Combine and save money and time!

What could go wrong without planning?

- Improper setup
  - One large customer installed LE R3 with default storage options and started running out of storage in IMS region when running a high volume of transactions
    - Even after fixing it in test, failed to move change to production
    - The storage defaults were changed in LE R4
  - One customer noticed a big performance hit when using TRUNC(BIN), problem was solved by using TRUNC(OPT).
    - TRUNC(BIN) should never be the default option
- Incomplete investigation into vendor products
  - One customer thought they were all ready to go to LE Release 3, but when they started testing they found that TCP/IP Version 2 does not run with LE, so they had to start migrating to TCP/IP Version 3 Release 1.
    - TCP/IP Version 2 was trying to dynamically access the C Version 1 library. TCP/IP Version 3 runs with either C Version 1, Version 2, or LE.

What do you get with COBOL/370?

- See ″Technical Advantages of LE/370″
  - In back of handouts

- For OS/VS COBOL users,
  - 31-bit capability
  - Much improved CICS interface
  - Structured programming constructs
  - Hex literals, reference modification
  - Re-entrant programs

What do you get with COBOL/370?

- For ALL COBOL users
  - The base for Object-Oriented COBOL
  - The base for Client-Server using COBOL
  - Any future performance improvements
  - New! OPTIMIZE(FULL) feature
  - New! RMODE option for NORENT above the line
  - Condition handling for entire application using COBOL programs
  - All I/O buffers above the line
  - Full data striping support (DFSMS)
  - Improved ILC performance
  - Intrinsic Functions
    - Including date functions that return 4-digit year dates

Preparation for migration

IBM

Take inventory of your applications

- COBOL runtime currently in use
- COBOL runtime options used
  - LIBKEEP? RTEREUS?
- COBOL compiler options used
  - RES? TRUNC? NUMPROC? CMPR2?
- COBOL source level 68, 74, 85, ?
- EDGE Portfolio Analyzer looks at load modules
- What, if any, will need to be re-linked?
  - For COBOL-PL/I ILC, start relinking before migration
    - See PL/I migration aid (in PL/I Migration Guide)
  - EDGE tool can tell you what needs to be linked
    - EDGE tool can build JCL to do relinking automatically

The most difficult migrations

- OS/VS COBOL NORES programs that use assembler with SVC LINK for ″CALLing″ subprograms
  - SVC LINK creates new enclave, OS/VS under LE/370 must be single enclave in non-CICS

- Any applications that do their own program management with assembler or have assembler programs that use high order bits
- Any applications that have assembler programs that use high order bits of registers

- Shops that use exits on the DCB EXLST for label processing
  - VS COBOL II and COBOL/370 use them all

- Any shop using modified ILBO* modules

- Applications that set SPIEs or STAEs and trap abends
  - Must migrate to LE condition handling
  - We have some relief for VS COBOL II programs that use SPIE under LE, as long as SPIE is RESET after application code

- OS/VS COBOL ILC with PL/I
  - Must upgrade COBOL before migrating to Language Environment

Using the new runtime

- Set up LE/370 for COBOL users

  If moving from VS COBOL II runtime:

  - Install CEEBX05A sample user exit
    - Instructions in LE Installation manual
    - Use CEEBX05A in place of CEEBXITA

  For all COBOL users, set default options:

  - ABTERMENC(ABEND)
  - CBLQDA(OFF)
  - TERMTHDACT(ABEND)

- For COBOL ILC, start relinking now

- For OS/VS COBOL with ESTAE exits, start source conversion

- For mixed-language shop, no COBOL ILC:

  For a COBOL-only migration

  - Install PL/I and C ahead of LE/370 in concatenation

Recommended options for all languages:

- ABTERMENC(ABEND) Default is RETCODE
- ALL31(OFF) (ON) if all migrated or on CICS
- CBLOPTS(ON)
- CBLQDA(OFF) Default is ON
- CHECK(ON) maps to SSRANGE of VS COBOL II
- NODEBUG USE FOR DEBUGGING
- RPTOPTS(OFF)
- RPTSTG(OFF)
- RTEREUS(OFF) RTEREUS can have nasty side affects!
- STORAGE(NONE,NONE,NONE,8K)
  - if no uninitialized variables
- STORAGE(00,NONE,NONE,8K)
  - will zero out WORKING-STORAGE variables (X′00′)
- NOTEST
- TERMTHDACT(DUMP) Default is TRACE
  - or TERMTHDACT(UADUMP) For SYSUDUMPs

TRAP(ON)

- - <u>TRAP(OFF) is only for diagnosis purposes</u>

ALL31(ON) is correct for OS/VS COBOL in CICS because OS/VS programs running in CICS do not bring up the LE/370 environment.

Only the VS COBOL II and COBOL/370 programs will be using LE/370. run in LE/370. COBOL uses no STACK storage and a less HEAP than C or PL/I. Working-Storage comes out of HEAP storage.

Testing your applications

- Test every vendor product and package first
- Test a cross section of your applications
  - You shouldn′t have to test every application
  - Test the difficult applications
  - Definitely test the critical applications
  - Test error conditions too: next foil
- Test in CICS, IMS/DC, DB2
- Try to re-create actual production use
- Use an entire system if possible
- Use Omegamon to zap LE into LINKLST for testing

Testing your applications

- Test missing DD statements for the following:
  - DISPLAY and ACCEPT
  - QSAM file - OPENed for OUTPUT and for INPUT
  - VSAM file - OPENed for OUTPUT, for INPUT, and for I-O
- Test PARM= string/ PROCEDURE DIVISION USING:
  - Containing no ′/′ characters
  - Containing a date - 08/16/95
- CALL ′user-abend-subroutine′
  - Your own
  - ILBOABN0
  - IGZEABN - believe it or not!

Testing your applications

- Test behavior of QSAM input and output files when an I/O error is encountered and no FILE STATUS checking is performed

- Test behavior of outputting more records than can be accommodated by the file allocation for a QSAM file with no FILE STATUS checking

- Test fixed file attribute conflicts between file label, DD statement, and FD for OPEN OUTPUT

- Test behavior of ADDing +1 to a USAGE DISPLAY signed and unsigned field containing blanks

- Test behavior of ADDing +1 to a PICTURE S999 COMPUTATIONAL data item that contains +999

- Test behavior of DIVISION by zero, for all data types, with no ON OVERFLOW clause

- Test behavior of MOVEing a USAGE DISPLAY signed and unsigned to a similar field if the sending field contains blanks

Testing your applications

- A suite of testcases covering this list of scenarios can be constructed with perhaps 15 or 20 source programs.

- In many cases, a single relatively simple test program can be used for several of the scenarios:

  - This can be done via JCL changes

    - Omitting a DD statement

    - Coding a conflicting DCB parameter on the DD

    - Coding a very small amount of primary file SPACE and no secondary SPACE

- In handouts from Rex Widmer GUIDE pitch, there is an example that can be used for multiple scenarios via changing the PARM = value

Moving LE/370 into production

- Install target depends on what languages are used
  - LINKLST or LPA only when everything runs under LE/370
- STEPLIB for old applications that will not run under LE/370
  - Complete the migration or re-write later
- STEPLIB provides maximum control with worst performance
  - Customer feedback says STEPLIB noticeably slower
- One IMS region at a time
- One CICS region at a time

Moving LE/370 into production

IBM

- Possible scenarios for controlled migration
  - Batch1:
    - Newly migrated applications STEPLIB to LE/370
    - Deferred applications STEPLIB to old runtime
    - At a given point change from old runtime to new in LPA/LINKLST
    - Start deleting STEPLIB for migrated applications
  - Batch2:
    - Change all applications to STEPLIB to current runtime
    - Install LE/370 in LPA/LINKLST
    - Deferred applications STEPLIB to old runtime
    - Delete STEPLIB to migrate each application

Moving LE/370 into production

- Scenario for controlled migration
  - Online (either IMS/DC or CICS):
    - Create new region that uses LE/370
    - Move transaction by transaction from old region to new
    - When all transactions moved, delete the old region
  - Online (TSO):
    - Use TSOLIB command to set up dynamic steplib concatenation

      (must be TSO/E 2.5 or later)
    - Under ISPF add LE to the ISPLLIB concatenation
    - If you can control the logon proc, put a STEPLIB there

COBOL source conversion

- OldBOL to NewBOL
  - OS/VS COBOL had LOTS of undocumented extensions
- 85 Standard or 97 Standard COBOL?
  - We can't wait until 1997 for OO COBOL!
  - We will also support the new standard when approved

Use tools!

- IBM COBOL Conversion Aid is bigger and better
  - Known as CCCA - 5785-ABJ
  - Over 80 user requirements added in Release 2

COBOL/370 compiler options

- **NUMPROC(MIG)**
  - **Should be default for most shops**
  - **Closest to OS/VS COBOL sign processing**

- **TRUNC(OPT)**
  - **Should be default for most shops**
  - **We have changed our recommendation for CICS and DB2**
  - **OS/VS COBOL NOTRUNC actually did truncate in some cases**
  - **TRUNC(BIN) only for specific programs**

- **OPT(FULL)**
  - **Don′t use for programs with flags in WORKING-STORAGE unless you add references to them**
  - **The following example shows how to code a PROCEDURE DIVISION reference that will keep BEG-MARKER from being deleted, but will not add any executable code to the object program:**
    - **The IF can never be true so the optimizer deletes the whole statement!**

```
.............   ....................................
....
....
........................................
   ...................
```

COBOL/370 compiler options

IBM

- If currently OS/VS COBOL using TRUNC as default or VS COBOL II using TRUNC(STD) as default:
  - Use TRUNC(STD) as the default under COBOL/370 or COBOL for MVS & VM.
  - They are identical!

- If currently OS/VS COBOL using NOTRUNC as default:
  - Use TRUNC(OPT) as the default under COBOL/370 or COBOL for MVS & VM.
  - They are very similar.

- If currently VS COBOL II Release 2 using NOTRUNC as default:
  - Use TRUNC(OPT) as the default under COBOL/370 or COBOL for MVS & VM.
  - They are identical!

- TRUNC(BIN) should NOT be used as a default option
  - It is TOO SLOW!

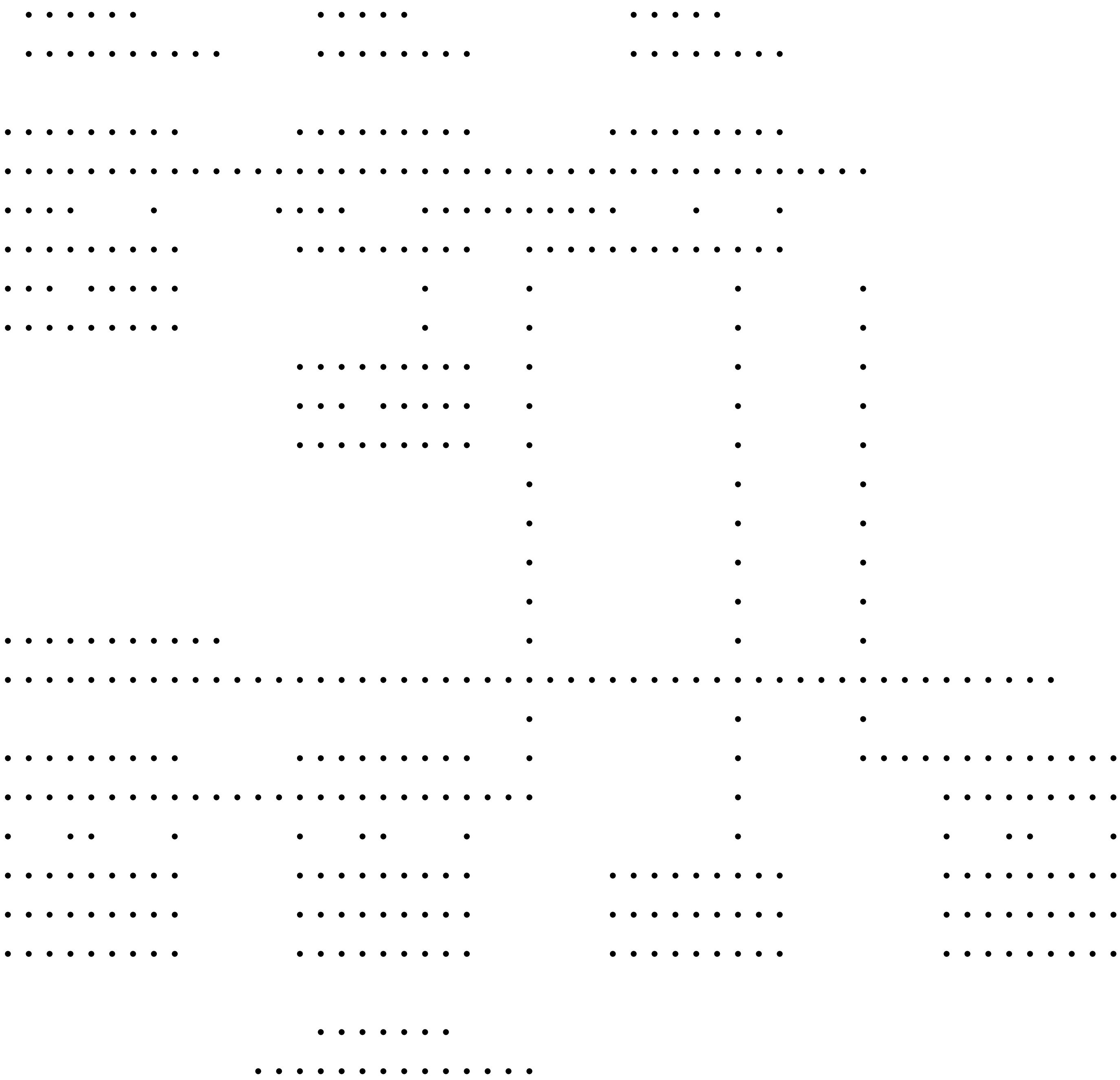
- TRUNC(BIN) should be used only for select programs that require guaranteed non-truncation of binary data.
  - This is particularly true if there is a possibility that data being moved into BINARY data items can have a value larger than that defined by the PICTURE clause for the BINARY item.
  - Example is PIC 9(7).9(2) that will have values bigger than 9,999,999.99
  - With TRUNC(BIN) it can hold up to 16,777,215.99

Adding new COBOL to old COBOL apps

How to change just one program, such as a date routine, to COBOL/370 in an OS/VS COBOL application

- A little about AMODE/RMODE
  - OS/VS programs are all RMODE=24 AMODE=24
  - COBOL/370 programs can be RMODE=24 or RMODE=ANY
  - COBOL/370 programs are all AMODE=31
- Are you ready for the 16meg line?
- Keep all data below the line until all source migrated
  - If RENT, then DATA(24)
  - If NORENT, RMODE(24) or RMODE(AUTO)
- I recommend migrating bottom subroutines first
- Old COBOL calling new COBOL is safer than new calling old
- Dynamic call is safer than static, mode switching
  - Can′t enter an OS/VS COBOL program with AMODE=31!
  - Get IKF995I ILBONTR AMODE ERROR.

Example: new calling old. All calls DYNAMIC



Adding new COBOL to old COBOL apps /Notes

The only CALLs that may have problems are from the RENT,DATA(31) programs calling OS/VS or NORENT (VS COBOL II). If they pass parms, then blam! Same problems if you replace the OS/VS COBOL with COBOL/370 or VS COBOL II w/DATA(24).

Hints and Tips

- Extremely short on-line transactions may have dramatic difference in performance.
  - Especially when coming from NORES environment
  - You may have to redesign if this is a problem
- Installation customization macros require SYS1.MACLIB
- Do not use default options without verifying that they are correct for your shop!
  - Too many options, and too many different requirements from customers
  - No way to have ″One set fits all″
- Keep your language products up to date on service as much as possible
  - IBM only tests with recent levels and
  - Down-level language migrations are harder
  - Example: VS COBOL II Release 4 base = VS COBOL II Release 3.2 at latest maintenance level
- Keep control of your language products usage
  - Do not allow programmers to chose what release, what maintenance level, or you will end up with a nightmare of products

What is the ″YEAR 2000″ problem?

2-digit year data is the problem

- **On 1/1/2000, the current year will be less than the previous year**
  - **Date field has only the last 2 digits of year**
  - **00 is NOT greater than 99**
- **Depending on the coding technique, you might not be able to tell that a customer had turned 18, or whether a balloon payment is due!**
- **Many businesses are experiencing problems already, as the turn of the century approaches**
- **There is no way to get 4-digit year dates using OS/VS COBOL or VS COBOL II**

The ″YEAR 2000″ problem

- There is no way to avoid the pain of changing databases and applications to use 4-digit year dates.

  - IBM does provide a short-term solution but it is inevitable that you must use 4-digit years at some point in the future.

- There are 3 ways to solve the problem using COBOL/370 and/or LE/370.

- Our short term solution is the ′century window′ feature of LE/370

  - Allows 2-digit years to be interpreted in a 100-year window

- IBM provides 2 long-term solutions. One is to use COBOL/370 Intrinsic Functions and the other is to use LE/370 callable services. Both of these methods return 4-digit year dates.

Long-term solution

IBM

- You must eventually:

   1. Re-write applications

      - Use Intrinsic Functions to get 4-digit year dates OR
      - Use LE/370 callable services to get 4-digit year dates

   2. AND Rebuild databases with 4-digit years

- The above are the long-term solutions that will take you and your company through to the year 9999

- However, there is a short-term solution that is easier

Short-term solution

IBM

Q: What if I can′t change all my applications and data before 2000?

A: Use the LE/370 century window in old applications to interpret 2-digit year dates as 4-digit year dates

- **Advantages:**

  Change only a few programs, no database change

  Cheaper, quicker, easier

- **Disadvantages:**

  This only buys you a few years, depending on the application

  You still must change all date programs and databases eventually

Q: Why can′t I user the century window forever?

A: It is a mathematical fact that a 2-digit year can only be unique in a given 100-year period. Over time you will need more than a 100 years for your data window - in fact, many companies need more than 100 years now!

The reason that the century window buys you time is that you know in any given application whether you are trying to figure out how old someone is (the date is in the past) or whether a due date has been reached (the date is in the future).

There are limitations. For example, the century window cannot solve the problem of trying to figure out how old a customer is if the customer is older than 100 years and you only have 2-digit year dates! For this problem and others you need COBOL/370 Intrinsic Functions or LE/370 date services.

LE/370 century window and old COBOL

Q: Can I call LE/370 services from OS/VS COBOL programs?

A: No, programs that need to call LE/370 callable services must be VS COBOL II or COBOL/370 programs

Q: How do I solve the YEAR 2000 problem in old applications?

A: Use the century window feature of LE/370:

1. Run your existing load modules under LE/370

2. Upgrade date routines to COBOL/370
   (or use dynamic CALLs from VS COBOL II programs)

3. Use century window of LE/370 to interpret 2-digit years:

   - CALL ″CEECBLDY″ USING input-date PICSTR
     LILIAN FC.

   - Results in an integer representation

4. Use Intrinsic Functions to obtain 4-digit year

   - DATE-OF-INTEGER gives YYYYMMDD

   - DAY-OF-INTEGER gives YYYYDDD

See example MAIN that calls SUBRTN

COBOL example MAIN

```
  ......... .........

............. ..... ........
           ............................
             ......................
                     ............................
           ....
             ...................
             ............
             .............
           ......

           ........

  .....    .................
             ............
               ......
                 ...............
               ..........
                 ...............................
             ........

           ............

  .................
```

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN

Year 2000: Integer Dates

- Problem: LE/370 Lilian and ANSI COBOL Integer dates are different
- Solution: Use LE/370 Lilian dates exclusively
- Problem: What about Intrinsic Functions?
- Solution: Use COBOL for MVS & VM Install option INTDATE
  - INTDATE(LILIAN) or INTDATE(ANSI)
  - Use INTDATE(LILIAN) to get LE/370 Lilian dates from Intrinsic Functions
- Question: What about ANSI Standards?
- Answer: There were too many Integer date formats even before LE!
  - PL/I - no integer date values
  - C - has integer dates that start with Jan 1, 1970
  - COBOL - has integer dates that start with Jan 1, 1601
  - LE/370 - has integer dates that start with Oct 15, 1582
  - CICS - has integer date that starts with Jan 1, 1900
  - MVS - has integer date that starts with Jan 1, 1900
  - DB2 - has integer date that starts with Jan 1, 1900

Why 2 different start dates for integer formats?

LE/370 and the 1989 ANSI Standard Amendment were developed at around the same time. The Language Environment group used a study published by IBM in 1984 that determined that the first reliable date we can count on is Oct 14, 1582. Every day after that date can be accounted for, while prior to that date there are ″missing″ days and months.

The ANSI COBOL committee X3J4 also decided to use an integer-style date, but they used the starting date of Dec 31, 1600. IBM gave them the results of our published study, but they chose to use their own starting date.

Upon further investigation, IBM found many different integer starting dates in use, and determined that LE should keep using the ″best″ starting date of Oct 14, 1582. Our default options for the COBOL for MVS & VM compiler must conform to the ANSI Standard, and thus the difference!

Integer Dates

- Recommendations:

  - Only use integer-form dates for performing date calculations in the same program. Convert to integer and back from integer in the same program.

  - Avoid storing integer dates in databases and files

  - Avoid passing integer dates from one program to another

- If you have a requirement to put integer dates in databases and files, then standardize on Lilian dates.

How long you can safely continue to run your OS/VS COBOL applications without a newer library is difficult to predict.

As long as nothing in your environment changes, you can run your OS/VS COBOL programs forever. However, if you change any other software in your environment, you could break any or all of your existing OS/VS COBOL applications.

A perfect example of this was DFSMS/MVS V1R3. The VSAM component changed in such a way that a bug in OS/VS COBOL library routine ILBOVOC was revealed. This bug is fixed in Language Environment for MVS & VM R5 but it still exists in OS/VS COBOL, and a future release of VSAM could again break all OS/VS COBOL programs in your shop.

Therefore, our recommendation is that even if you decide to buy the IBM offering that gives you an extension on supported service until June 1996, you should start migration to COBOL/370 and LE/370 as soon as possible.

PERFORMANCE:

- Inter-language CALLs are faster under LE/370 between COBOL and other languages, with about 80% less overhead for each call.
- COBOL-assembler-COBOL calls 80% less overhead than VS COBOL II.
- All COBOL calls are faster under LE/370 than under VS COBOL II.

  (In some cases, DYNAMIC CALL is faster than under OS/VS COBOL!)
- COBOL/370 compiler changes for object program performance: (Improvements over VS COBOL II, only available under LE/370)
  - OPTIMIZE(FULL) option
  - Static initialization of WORKING-STORAGE variables at compile time for NORENT programs
  - Optimized parameter list generated code (for USING phrase)
  - Improved performance of variable-length MOVEs
  - Optimized code generated for main entry point
  - Dynamic CALL literal performance improved

Technical advantages of LE/370

IBM

NEW FUNCTION:

- Support for OpenEdition MVS services, including support for some of the C language functions defined in the POSIX standard
- ANSI 1985 Intrinsic Functions available in COBOL/370
- YEAR 2000
  - LE/370 callable services that interpret 2-digit year dates as 4-digit year dates. This solves the year 2000 problem prior to changing the database to 4-digit year dates. It also makes it easy to write an application to change the database.
  - COBOL/370 Intrinsic Functions return 4-digit year dates

    (no 4-digit year dates available on OS/VS COBOL or VS COBOL II)
- High-precision math routines available to any language

  (LE/370 callable services are only available to LE-conforming languages or LE-conforming assembler)

NEW FUNCTION:

- **LE/370 Condition Handling (abend intercept, "0C7 catcher", etc)**

  Allows application-level condition handling so that each routine can be written without dealing with error conditions. This allows routines to be written for success and performance rather than written to address errors in individual statements. The error- handling routines can be written in the High Level Language of choice, thus helping you move away from assembler.

  (Before LE/370 the error-handling code had to be in the same language as the application, or in assembler language. Also, in COBOL you could only have statement-specific condition handling, which means each new program had to have its own condition handling and the associated performance overhead.)

NEW FUNCTION:

- Object-Oriented COBOL on MVS requires LE/370
- Client-server using COBOL on MVS requires LE/370
- SOM support on MVS requires LE/370
- C++ for MVS requires LE/370
- DB2 V4 Stored Procedures function requires LE/370
- CICS V4 Autoinstall Exit written in COBOL requires LE/370

PLATFORM OF THE FUTURE:

- **Any future performance improvements or language enhancements such as DSOM support or XOPEN COBOL will only be available under LE/370.**

- **LE/370 will be a basic part of MVS, VM and VSE through OS/390 and BOS/390, and will be mirrored on OS/2 and AIX as well. LE/370 is already built in to the OS/400 operating system.**

- **More support of the POSIX standard and possible exploitation of the XPG4 OPEN standard would allow users access to a UNIX-like environment within MVS using C and possibly COBOL.**

SYSTEM EXPLOITATION:

- QSAM buffers can now be above the line for full exploitation of DFSMS and data striping.
- COBOL EXTERNAL data can now be above the 16 MEG line for COBOL/370

  (Had to be below the line in VS COBOL II)
- CBLPSHPOP run-time option allows choice of performance versus error handling. (Under VS COBOL II the user had no choice and always took the performance hit of PUSH HANDLE and POP HANDLE for CALLs)
- New COBOL/370 compiler options:
  - RMODE option allows NORENT programs to run above the line
  - OPT(FULL) automatically optimizes out unused data items
- New LE/370 support in R3:
  - Allows COBOL/370 programs compiled with NORENT and RMODE(ANY) to run above the 16 MEG line. (Only RENT was supported above the line under VS COBOL II)

USEABILITY:

- **All language error messages now in latest CD-ROM collection kit**

  (Previous to LE/370, messages in 3 different manuals)

- **CBLQDA run-time option prevents dynamic allocation of QSAM files**

  (Only available as a User Mod under VS COBOL II)

- **COBOL/370 removes restrictions for APPLY WRITE-ONLY**

  (Under OS/VS COBOL, had to use FROM on WRITE. Also, the subfields of the records could not be referred to in procedure statements.)

- **Storage tuning via run-time options (no link-edited assembler, which was the only way to tune storage usage under VS COBOL II)**