The Year 2000, COBOL programs, and You!

GUIDE session AP152
Tom Ross

Language Products
IBM Santa Teresa Laboratory

April 1997

Outline

- What is the Year 2000 problem?
- What is IBM doing about it?
- How to solve the Year 2000 problem
- Assessment and planning
- What does ″Year 2000 ready″ mean?
- Year 2000-ready products
- Y2K conversion and language migration
- How to make your applications Y2K ready
- Testing your applications
- Make sure vendors are on board
- How to get more information
- DFSORT and the century window
- COBOL program samples

What is the Year 2000 problem?

- Date information stored with only 2 digits
- Only 140 weekends left to roll out changes!
- Incorrect century
  - 04 assumed to be 1904
- Arithmetic calculation
  - A person born in 1950 calculated to be -50 years old
  - Due date of 30 year mortgage started in 1990 could be calculated to be 1920, paid off!
- Sequence
  - 12/31/99 comes after 01/01/00
  - Credit card that expires in 00 considered expired!
  - IMS database updates will fail without V5 and fix
- Leap Year
  - If divisible by 100, then not a leap year, right?
  - Except 2000 is a leap year

What is the Year 2000 problem?

For non-application code:

- **Archiving and deleting of old datasets**
  - **They might not be old on Jan 1 2000!**
- **Conventions of '99365' meaning keep forever**
  - **That date is only 3 years and 40 days away**
- **Expiration dates on tape labels**
- **Homemade dataset managers**
- **Any and all system software that has not solved the problem**

What is IBM doing about it?

- Awareness and communication
  - Year 2000 Planning and Implementation Guide
  - Year 2000 Fact Sheet
- Planning and support
  - Year 2000 Technical Support Centers
- IBM "Year 2000 Ready" products
- Consulting & Services
  - Global Services Division
- Tools & Solutions

How to solve the Year 2000 problem

1. Replace non-″Year 2000 ready″ hardware

2. Replace non-″Year 2000 ready″ system software

3. Fix all of your applications and data

4. Link and run applications with

Language Environment for MVS & VM

5. Test

That′s it!

But seriously folks

- Get executive buy-in
- Establish a team
- Create a plan
- Make an assessment of your whole operation
  - IS systems
  - Desktop systems
  - Elevators, vaults, anything electronically controlled
  - Satellite communication!
- Make an assessment of your applications
- Link and run applications with

Language Environment for MVS & VM

- Fix any Year 2000 problems in applications
- Unit Test
- Install "Year 2000 ready" hardware
- Migrate to "Year 2000 ready" system software
- "Time Machine" test

The most important phases of the whole process!

- You may not be able to convert all applications in time
  - Maybe not enough programmers in world to fix all applications
- Determine which ones are most important in each business area:
  - Payroll, Accounts Receivable, HR management, Billing, Inventory, Manufacturing, Purchasing, Tax, Investments, Travel Expenses, Bldg Maint, etc
  - Which are Mission Critical?
- Identify which applications can be fixed after Year 2000
  - Include plans to get along without them until they are fixed
- Identify which applications MUST be fixed before Year 2000
  - Create priority list, fix them in order
  - This way, even if you don′t complete the job, you maximize benefit

What does ″Year 2000 ready″ mean?

Year 2000 ready Product

The Product, when used in accordance with its associated documentation, is capable of correctly processing, providing and/or receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software and firmware) used with the Product properly exchange accurate date data with it.

- **This can be applied to your applications as well**

- **More simply, ask yourself:**

  - **Will your applications run after 1999?**

  - **Will they produce the correct output after 1999?**

Year 2000 ready products

- See the supplied book, GC28-1251
- Make sure that your hardware is ready
- Make sure that your system software is ready
- Examples:
  - Only MVS V5 or later is considered Year 2000 Ready
  - CICS V3.3 or earlier will not run across midnight 12/31/1999
    - Might be able to do a complete ice-cold rebuild/restart for V3.3
  - IMS V4 and earlier will fail the first database update of 2000
    - 2-digit year dates in IMS Log records
    - IMS V5 with maintenance uses century window
    - IMS V6 will use 4-digit years
  - OS/VS COBOL and VS COBOL II are NOT Year 2000 ready

Year 2000 ready products

IBM

- Languages are a whole different animal
  - Non-ready System Software will fail regardless of your applications
  - Non-ready languages will continue to run after 1999
  - Year 2000 ready language provides 4-digit year dates
- For COBOL, only COBOL for MVS & VM and Language Environment are ready
  - NO 4-digit year date COBOL language in:
    - OS/VS COBOL
    - VS COBOL II
- Your applications need century window and 4-digit year date functions
  - With only 2-digit year dates and no window, then fail!
- Do your applications need to be changed?
  - 90% of financial applications need to be changed
  - 75% of other applications need to be changed

Year 2000 ready products

- There are only 2 problems to be concerned with for COBOL applications and the Year 2000:

  1. Date Related Logic

  2. Service support from IBM

- Date Related Logic problems

  - Must be fixed

  - More later

- Service support from IBM

  - OS/VS COBOL service support ended June 1994

  - VS COBOL II service support will end soon after 2000

    - (Announcement in March 1997)

  - You can run old programs indefinitely without service extension!

Migrate without recompiling!

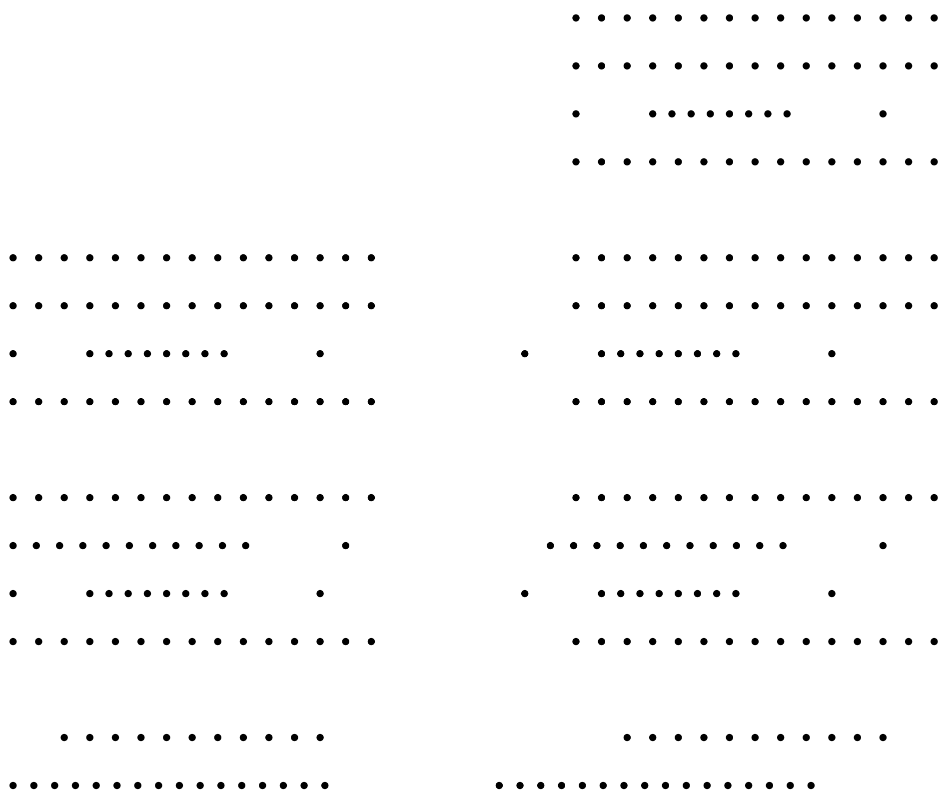
Year 2000 ready products

- Service support for OS/VS COBOL and VS COBOL II programs
  - Available indefinitely by using Language Environment for MVS & VM
  - Just a run-time migration away!
  - OS/VS COBOL compiler not supported, but compiled code will be

- OS/VS COBOL applications with NO DATE RELATED LOGIC
  - Could be run fully supported by IBM beyond 1999
  - Again, a run-time migration only
  - No source changes or recompiles necessary

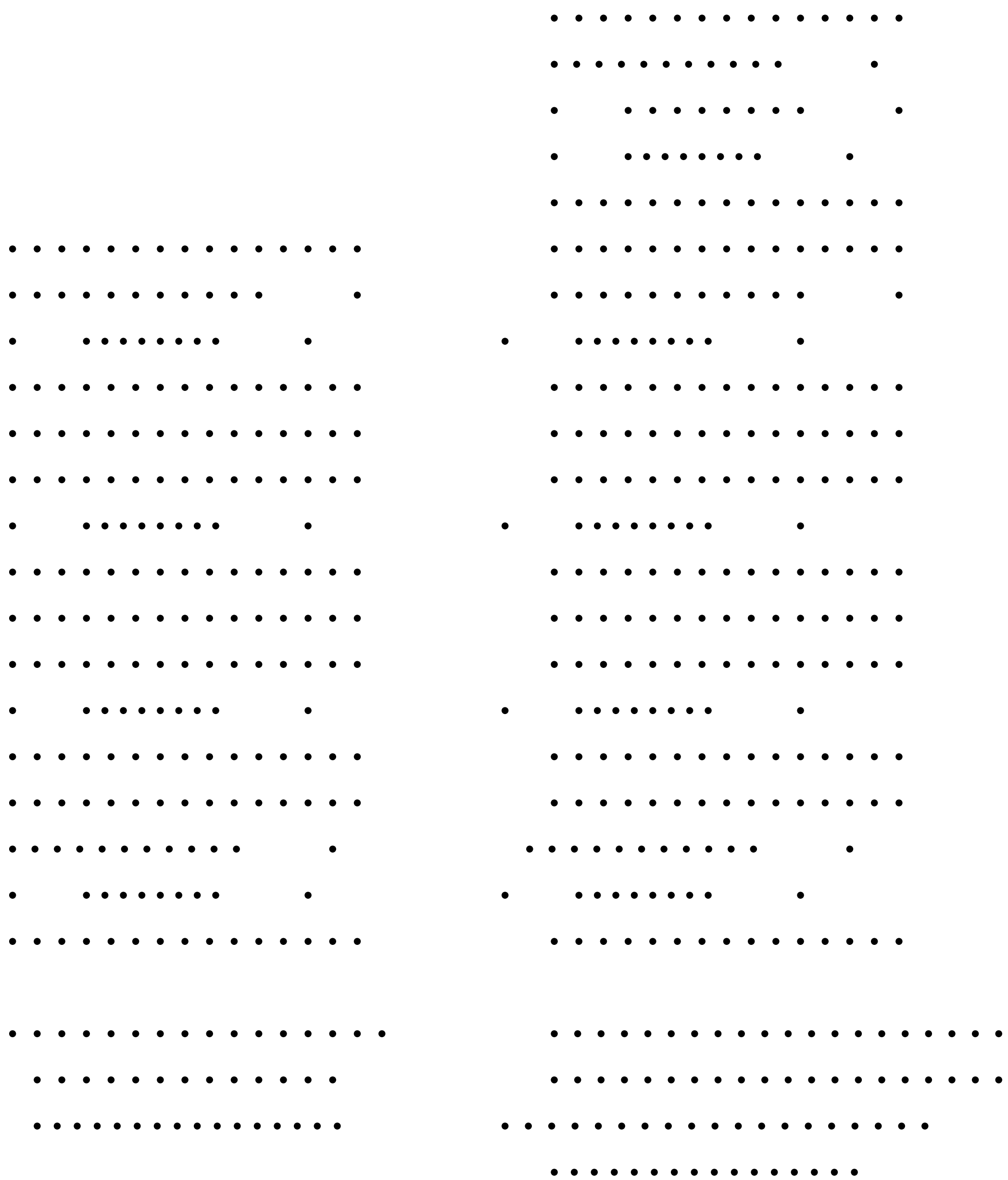
- OS/VS COBOL applications WITH Date Related Logic
  - Must be examined for possible Year 2000 problems
  - If problems, then the programs will have to be changed
    - Which means at the very least a compile, link and test.
    - At this point you can combine source upgrade (compiler migration) with Year 2000 conversion.
  - Only VS COBOL II or later programs can use IBM services to obtain a 4-digit year date

Obsolete/Stabilized run-time libraries



- Old languages run-time library had same name as compiler
- Had to separate compiler and libraries for Language Environment

Strategic Run-time libraries



LE R3 End Of Service in 3/97

LE R4 End Of Service in 9/97

Programs supported by LE R5 and later

Y2K conversion and language migration

- 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!

- ″Collateral Benefit″

Four ways to fix the problem

1. Field expansion
   - Expand 2-digit years to 4-digit years
2. Century window
   - Interpret 2-digit years as 4-digit
   - Sliding or fixed window
3. Century encoding/compression
   - Pack 4-digit date fields into 2-digit size field
   - Several ways to do this, hex, packed, etc
4. Integer format dates
   - 4 bytes to hold YYYMMDD

1. Field Expansion: Best solution

- You must eventually:
  1. Re-write applications
     - Use COBOL Intrinsic Functions to get 4-digit year dates OR
     - Use LE callable services to get 4-digit year dates
  2. AND Rebuild databases and files with 4-digit years
- The above are the long-term solutions that will take you and your company through to the year 9999
- Advantages:
  - Straightforward code changes
  - Minimum testing, maybe no "Time Machine" testing
  - Fastest resulting code
  - Permanent fix
  - Cheaper maintenance
- Disadvantages:
  - Change databases and files

2. Century window: Short-Term solution?

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

- Advantages compared to field expansion:
  - No database or file changes

- Disadvantages compared to field expansion:
  - Performance will be slower with the increased logic
  - More testing to validate changes
  - Very difficult to manage across applications
  - Translation could be wrong
  - MUST do "Time Machine" testing
  - Code changes are more error-prone, require more expertise
  - Increased maintenance costs
  - This only buys you a few years, depending on the application
  - You still MUST change all date programs and databases eventually to use 4-digit years

3. Century encoding/compression

- **Advantages compared to field expansion:**
  - **>>No database or file EXPANSION**
- **Disadvantages compared to field expansion:**
  - **Performance will be slower with the increased logic**
  - **More testing to validate changes**
  - **Very difficult to manage across applications**
  - **Translation could be wrong**
  - **MUST do ″Time Machine″ testing**
  - **Code changes are more error-prone, require more expertise**
  - **Increased maintenance costs**
  - **>>This only buys you a few years, depending on the METHOD used**
  - **You still MUST change all date programs and databases eventually to use 4-digit years**
  - **>>Must translate whenever externalized**
  - **>>Can′t read in dumps and files**

4. Integer format dates

- Advantages compared to field expansion:
  - >>No database or file EXPANSION
- Disadvantages compared to field expansion:
  - Performance will be slower with the increased logic
  - More testing to validate changes
  - Very difficult to manage across applications
  - Translation could be wrong
  - MUST do ″Time Machine″ testing
  - Code changes are more error-prone, require more expertise
  - Increased maintenance costs
  - >>Must translate whenever externalized
  - >>Can′t read in dumps and files

Window vs Field expansion: which is better?

- Not only will it not work forever:
  - It will cost more to maintain
  - You will have to do it all again later!

Century window: what′s it good for?

- Use the century window to expand your data
- Create mirror image of production system
- Read in the old, write out the new

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

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

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

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

- Do the same thing for IMS and DB2 data
- DFSORT can do the same thing

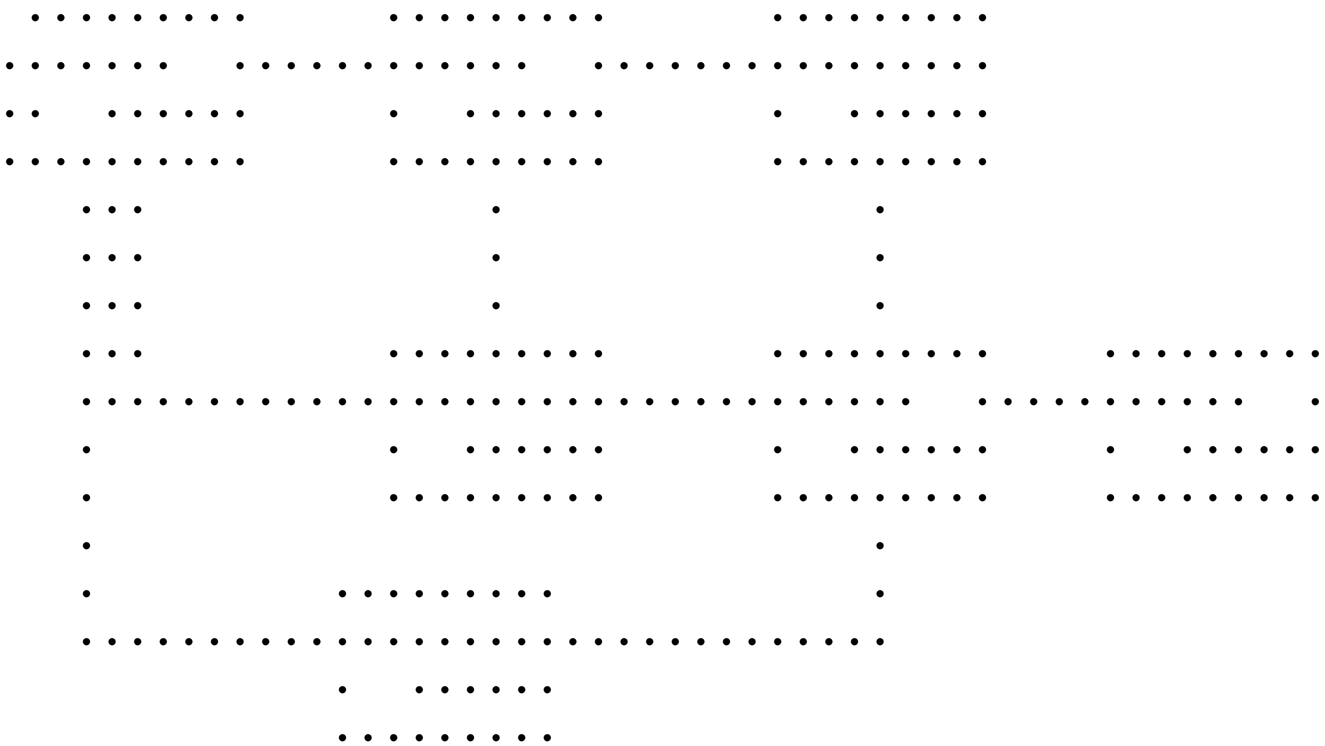
More on integer format dates

- Provided by COBOL Intrinsic Functions and by Language Environment date/time callable services
- Number of days from some point in the past
- PRO: Use only 4 bytes to store a date
- CON: Too many Integer format starting dates:
  - CICS, MVS, DB2 - start with Jan 1, 1900
  - PL/I - no integer date values
  - C - start with Jan 1, 1970
  - COBOL - start with Jan 1, 1601
  - Language Environment - start with Oct 15, 1582
- Solution for COBOL: Use COBOL for MVS & VM Install option
  - INTDATE(LILIAN) or INTDATE(ANSI)
  - Use INTDATE(LILIAN) to get LE Lilian dates from Intrinsic Functions

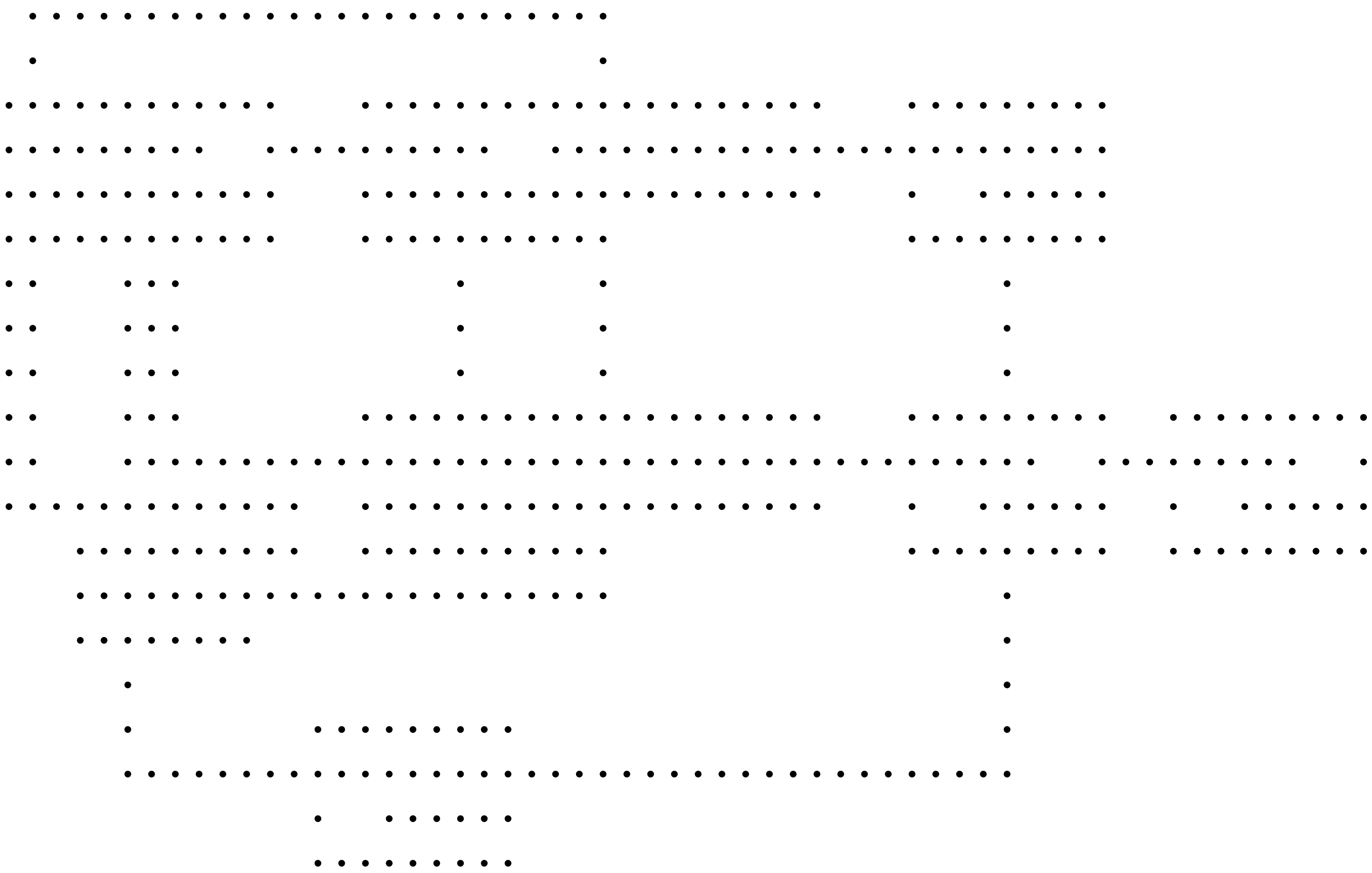
What to do with historical data with field expansion method?

- Basically just two choices for data:
  - Expand all old files/databases
  - Keep copies of old data
- Which leads to choices for code:
  - Copies of old code to access the data as well or
  - Code needs to handle both 2-digit and 4-digit years
    - Could use conditional "bridges" when accessing old data

Partitions and bridging



Partitions and bridging



- **Only changed APP1, APP2, DATA1**
- **Burn bridges after DATA or APPS changed**

Testing your applications

- Regression test
  - Current production data
  - Regular test environment
- Year 2000 Unit test
  - Date simulators are enough for application testing
    - TicToc, HourGlas 2000, Xchange, etc
  - DO NOT INSTALL DATE SIMULATORS ON PRODUCTION COPY OF LE
    - They all corrupt date/time module of Language Environment
    - LE date services do not use SVC 11
    - Use a separate dataset for LE testing vs LE production
- Other future test
  - Data with future dates

″Time Machine″ test

- ″Time Machine″ is a machine IPLed to future date
- Time Machine testing is the LAST STEP in the process
- Keep future date machines separate from others
- Do not allow future dates to corrupt:
  - Networks
  - Datasets
- Ideal Time Machine: PC Server 500 System/390
- Think of Time Machine as black hole
  - Can move data, programs, files in
  - Don′t let anything escape with dates intact!
  - May be able to move files out on unlabeled tapes

MVS LPAR as a Time Machine

- Create LPAR
  - Connected to production DASD
  - Connected to LPAR unique Time Machine DASD
- IPL to current date
- Copy programs and data over for testing
- Disconnect LPAR from production
  - DASD, TAPE, networks
- IPL to future date
- Test the system!
  - Watch for "expired" software licenses
  - May need to have special agreements with vendors
- Create fixes in current-year system
  - Move to Time Machine on tapes, "Sneaker NET"

Make sure vendors are on board

- 25% of companies will go out of business because of Year 2000
  - Don′t let yours be one of them
- Make sure your software vendors are fixing their Year 2000 problems
  - If not, you need to plan for replacements!
- Make sure all of your companies suppliers are also fixing their Year 2000 problems
  - Remember the GM brake division strike?

How to get more information

IBM

- GC28-1251: The Year 2000 and 2-digit dates:

  A Guide For Planning and Implementation

  - The manual can be downloaded from the IBM Year 2000 website
  - http://www.software.ibm.com/year2000/index.html

- Year 2000 Technical Support Center

  Ask questions via:

  - YEAR2000 CFORUM
    - (Available though TALKLINK of IBMLINK)
  - Y2KTSC@vnet.ibm.com

- Year 2000 Testing by S390 group

  - http://ppdbooks.pok.ibm.com/cgi-bin/bookmgr/

    bookmgr.cmd/books/E0S1P606/contents

Non-IBM Year 2000 web sites

- http://www.source-recovery.com
  - Recover COBOL source from load modules
  - Assembler too!
- http://www.year2000.com (De Jaeger)
  - General info on Y2K issues
- http://www.ssa.gov/year2000/y2klist.htm
  - US Gov list of Y2k Ready software
- http://www.mitre.org/research/cots/compliance_cat.html
  - DISA Year2000 COTS Product Compliance Catalog
- http://www.gmt-2000.co.uk/gmt-2000/
  - Company has done testing of PC Software
- http://www.cinderella.co.za
  - Another company that has done testing of PC Software

DFSORT and the century window

- Apply fix for APAR PN71337 (PTF UN90139) to DFSORT R13
- To use the century window function in a COBOL SORT you need to tell DFSORT when your century window starts and which fields are to be sorted using the century window
- Use the following 2 control statements in JCL:

  1. OPTION Y2PAST
     - OPTION Y2PAST=nn
       - Where nn=number of years back to start of window
     - OPTION Y2PAST=nnnn
       - Where nnnn=fixed window start date
  2. SORT FIELDS=(n,n,Y2C,A,...)
     - Where Y2C, Y2Z, Y2P, and Y2D identify 2-digit year fields in character, zoned decimal, packed decimal, and decimal form

- For more information, see DFSORT documentation

www.storage.ibm.com/storage/software/sort/srtmhome/htm

DFSORT and the century window

- The OPTION statement can be coded in the IGZSRTCD DD statement

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

- The SORT control statement must be coded in a DFSPARM DD statement
- To SORT records that start with mm/dd/yy as yyyymmdd:

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

- Other notes on SORT:
  - Use a local name if your installation changed the default name DFSPARM
  - A SORT statement can not be specified in IGZSRTCD
  - This will override the SORT FIELDS produced by the compiler, so you must make sure that you identify all fields that are to be sorted.
  - These are the ASCENDING/DESCENDING KEYS in the SORT statements in your COBOL program

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

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

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

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

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

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

COBOL example MAIN

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

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

           ........

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

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

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

COBOL example SUBRTN

COBOL example SUBRTN

COBOL example SUBRTN