ソフトウェア工学 114－11
（１９９７．５．２３）

パソコンで稼働する西暦２０００年問題対応ツールの開発

大林　久人
東京情報大学

本学での２０００年問題への対応にパソコンを利用することを前提として筆者はここに紹介するツールをメソドロジーとともに開発した。
結果として、パソコンの利用によりホストの負荷を軽減すること、複数台のパソコンによる並行作業でスケジュールの調整を楽にできること、作業の単純化によりバイトの利用を可能にすることなどが実現できそうである。

Tools for solving "the Year 2000" problem on PC

Hisando Obayashi
Tokyo University of Information Science

We developed software tools working upon PC, for solving the "year 2000" problem in our university. We discuss the way to solve this problem using these tools.

1　はじめに

西暦２０００年問題は大学と無縁のものではない。１９９９年まで残された時間は２年、本学の電算センタが管理する事務系の処理のためのソフトウエア資産は約３５００本、ほかにＪＣＬが１１００本ほどあり、その９割はもともと交代した派遣スタッフによる産物である。それへの対応をを２名の専任職員と数名の現在の派遣スタッフが抱える学年進行中の新学科のためのシステムの見直しとソフトウエアの修正、通常の保守作業に加えて負担させることは到底無理と考えられた。

期間内に対応をすませようとすれば数千万円の外注費の増加は避けられないとして問題は対応できる要員が確保できるかどうかである。現時点での外注依存はあまりにもリスクが多い。情報サービス産業協会「ニュース速報 No.299 」(1996.12.25)には1997年以降の情報サービス産業に対する 187万人月の注文に対して62万人月が不足することを予測記事として掲載している。メーカ、大企業と官公庁からの2000年対応作業の発注量しだいで中小ユーザからの注文には応じきれないという予測である。

幸い本学のソフトウエア資産は外注とはいえ、派遣者の手をかりた内製ソフトのためドキュメント類もほとんど散逸せずに残っている。要員の不足は、ツールの使用とバイトの利用で補うことにして自力対応の道を選んだ。また、設備は校内に４００台以上あるパソコンを利用し、テストだけは夏季休業期間中にホストで行うことになろう。

今回、発表するパソコンで稼働する2000年問題対応のツール群はそのために開発したものである。

なお、本学のプログラムはすべて COBOL言語を使用して作成されており、それ以外の言語に関するツールの開発は見送った。

2　対応方法

西暦2000年問題の原因は、年を表すのに西暦下２桁だけを使ってきた扱い方がまずかったというだけのことである。ユーザの側の問題を根本的に解決する方法は、ファイルやデータベースに含まれる日付項目をすべて８桁にして、年を４桁で表記するように変更することであるが、そのための作業量は日付を現在のままの６桁で扱う内部対応方式に比べると余りにも多くなりそうであった。

本学では出力リストや入力画面上の日付表記は原則として変更しないことで４割程度のプログラムはほとんど手を加えずにすむ見通しから内部対応方式で進むことにした。また、データの配列が混乱する問題はメーカからＩＢＭ社のＤＦＳＯＲＴ相当の汎用ルーチンが提供された時点でソートパラメータを修正するように準備しておき、提供されなかったときはCOBOL の整列機能を利用して専用プログラムを学生に作らせることで解決する予定をたてている。ただし、内部対応方式は問題を先送りするだけのものであることは認識している。

とにかく年の４桁表記を実現するには、ほとんどすべてのデータファイルの桁数引き延ばしの作業とともにそれに対応するプログラム中の PIC句の変更、JCL やVSAMファイルのDEFINEの変更、ソートパラメータの書き変えなどが必要になる。期間計算などの必要に応じて、すでに上２桁に19を補って処理していたプログラムは、余分なコードを取り除く。データファイルの桁数変更は、簡単なユーティリティで処理できるものではなく、結果的にはファイルごとに１本ずつのプログラムを作りそれをコンパイルして実行することになろう。４千本のプログラム、１００種類のファイルがあるとすれば、４千本のプログラムについては桁数の変更・余分になるコードの除去などの作業が必要になり、形式の変更される百種類のデータファイルのために１回限りの専用プログラムを百本作成し JCLを作って実行する。その上で修正済みのプログラムをリコンパイルし、テスト用のデータファイル（日付を1999年から2000年にずらして設定したファイル）を用意してすべてが現行通りに処理できるかどうかを確認する。単純作業が中心になるとはいえ、残された２年ほどの短期間にすべての作業を終了させるには、要員の確保とともに変換作業に使うマシン時間と新旧２本立てとなるファイルの格納スペースなどの問題も解決しておかねばならない。

3　本学での対応作業と使用ツール

本学での対応作業に備えて、すでに開発をすませたツールは次のとおりである。これらのツールを使用しながら以下に説明する手順で作業を進める予定である。

○棚卸し段階向けのツール

ZPROJCL	JCL からのプログラム･ファイル･ソートパラメータなどの抽出用
ZPJCLST	ファイルと使用プログラムの一覧データを作る
ZPJCLPRG1	プログラム名をJCL 順に並べた検索指示用のデータを作る
ZPJCLPRG2	ソートフィールドを並べたリストを作る
ZPFILE	COPY CALL文およびファイル名とOPENモードの検索用データを作る
ZPCPLST	COPY CALLの対象ファイルと入出力ファイルの一覧データを作る
ZPCPYPRG	COPY ファイル名を並べた登録ファイルを作る

○調査段階向けのツール

ZPRO4　文字列検索 COBOL プログラム用

ZPRO1　文字列検索 COPY句で使うファイル用

ZPLOGMK　検索された行にマークをつける

ZPDTNAM　検索された語とPIC 句の一覧をDTNAME.REGに累積する

ZPDTCHCK　レコード記述をもとに日付らしい値を持つフィールドを探す

ZPCONVFM　簡易なリポジトリを作りやすくするため型式変換をする
　リポジトリは簡易データベースソフトを使って作成する

○修正段階向けのツール

ZPROLNG　プログラム中の字数書換え

ZPRODAT　データファイルの形式変換

DATDIVI　同上　ファイルごとの処理プログラム作成

ZPROINS　標準コードの引き込み用

ZPINSCHK　標準データ名との重複チェック

ZSTRCHNG　データ名などの変更　文字列の書換え

○テスト段階向けのツール

ZPTESTDT　テスト用に日付を先にスライドしたデータファイルを作成

対応作業は、まず、ソフトウエア資産の現状把握からはじめる。サブシステム別にソースプログラム・コピーファイルなん本のような統計はとられているが、ホストのライブラリ中に実際に存在するプログラムの棚卸しは不可欠であろう。本学でも現用の JCLに含まれるプログラムのソースを確認し、ライブラリに存在する残りのプログラムが不要な資産かどうかを調査するところから着手する。

20年も経過してライブラリに存在するプログラム本数が 10000本をこえるようになった大規模ユーザでは、複数のライブラリが存在するためプロダクションライブラリにあるべき現用プログラムのソースやコピーファイルがプログラマ個人のライブラリに残っているようなケースが多く、所在を確認するだけの作業に多くの時間を浪費することが多いといわれている。本学のように開校いらい十年程度しか経過していないユーザの場合は、開発途上のサブシステムが多く、所在不明のソースプログラムなどの問題は少ないかわり開発の一時凍結と対応作業のスケジュール調整が問題となった。

ホストに存在するJCL の一覧をリストおよびファイルに出力させてから、１本ずつのJCL をサーバ経由でパソコンにダウンロードする。この作業はパソコンにあるようなファイルハンドラを使って連続的に行いたいが、ホストでは無理のようなので単純な反復作業を行なうことになる。このJCL から、プログラム名、使用ファイル、ソートプログラムにあるソートフィールドなどを抽出するツールが最初に使用するツールである。抽出したプログラム名をもとにホストのライブラリからプログラムソースを１本ずつダウンロードし、そのプログラム名を登録したファイルを使って、ソースの揃ったサブシステムから、プログラム中に記述されたコピー句で参照するファイル名やCALL文で呼び出す副プログラムなどを抽出する。これらは JCLから抽出できないからである。コピーファイルのダウンロードも続きの作業となる。この間に参照されなかったソースファイル、コピーファイルがあればそれについての調査を予定している。

次の調査段階の主要な作業は、全プログラムを対象とする日付項目の検索とその波及調査および演算、大小比較などを行う場所の検出である。日付項目の調査は、基本的には悉皆調査を必要とする。その結

果、もし内部対応すべきプログラムが 100パーセントに近いときは、データファイルの桁数引き伸ばしで対応するように方針を変えざるを得ない。作業の量と質を考えると単純にデータ項目を引き伸ばすほうが、ロジックに修正を加える部分が少ないだけ有利といえる。

プログラムリストから日付項目を洗い出すときの問題点は検索する文字列が多すぎることであろう。幸い、本学のように開始時期からすべてのファイルの記述はコピー句によるプログラムへのCOPYを守らせてきたケースでは、かりにHARAI SIHARAI のような項目名があって、SIHARAI = 支払額、HARAI = 支払日のような関係であったとしても、すべてのプログラムに共通の変数名となる。しかし、作業用の変数につける名前はプログラマの個人差がおおきく、英語あり、ヘボン式ローマ字、文部省式ローマ字ありの状態である。たとえ命名規約を作り、条件書ですべての変数名を指定したとしても、プログラムに１００パーセント反映できるかは疑問である。

そのため、本学でもとりあえず考えられる文字列を使って２次波及先まで抽出するツールによる検索を行い、その結果とシステム設計書、プログラム条件書に記載されている処理との整合性のチェックをサブシステムごとのサンプルプログラムについて行い、全体の検索処理の前に検索すべき文字列の調査をすませたい考えである。このツールは検索したい文字列を実行中に追加するとパソコンのテキストファイルに追記するように作成してあるので、チェックを繰り返しながら検索すべき文字列を増加させていくことができる。それでも確証が得られない場合に備えて、実際のデータファイルの内容を数百件程度抜き取り、日付らしき値を持つフィールドをチェックするツールも別に開発した。数回の保守作業を重ねて手元にあるファイルレイアウトも信用できない場合を想定し、現在そのファイルをデータとして処理していることが明確なプログラムのレコード記述を利用して、そこにある項目ごとに日付らしい値を持つかどうかを検査するツールも用意した。

また、検索された項目名については、それぞれの字数、使われているプログラムなどを参照できるように、一つのファイルに抽出していくツールを用意した。このファイルの内容をパソコンで作動する表計算ソフト、簡易データベースソフトなどに入力して簡単な辞書を作成するためである。表への入力を容易にするためレコードをCSV 形式やK3形式に変換するツールは以前から作成してある。

修正段階では、データファイルの形式変換とプログラム・コピー句の字数増減をするためのツール、内部対応のために標準ルーチンを挿入するツールなどを使用する。日付項目は６桁のままとするが１９９８年草々には新郵便番号への切り換えを行うことを予定している。これらはコピー句にあるレコード記述を加工して桁数増減のパラメータを作成し、それによって処理を行うツールを使用する。

標準ルーチンは、たとえば６桁の日付同士の比較を行う場合、それぞれを８桁のワークエリアに移してから比較するルーチンに置換する形式で挿入する。パターンは簡単なので、習熟すれば学生でもかなりの部分は対応できよう。いくつかのルーチンを先行して作成したが、調査段階で検索した結果から日付を使う演算のパターンを整理しそれにみあう標準ルーチンをファイルに追加してカストマイズする。挿入用のツールとは別に標準ルーチンの使う変数名との重複をチェックするツールおよび重複した変数名を変更するツールも準備した。変数も定数も新旧形式で変更するもので、消費税率の変更にも利用できる。

4　パソコン使用の利点

低価格でそこそこに能力があると評価されるパソコン、冷静に考えれば、現在世界の潜在コンピュー

ティングパワーの５割以上を占めるといってよい。ビジネスシステムの中ではデータの入力端末としての役割しか与えらないことが多かったかもしれないが、計算能力にしぼれば20年前の技術計算むけ大型コンピュータなみの能力を発揮できる。ファイル媒体にしても、30年前の磁気テープ５巻に実質的に格納できた情報は、ＭＯ１枚に満たない程度であった。

パソコンではプログラムのファイル名をひとつのファイルにまとめて記録しておき、読みだしながら対象ファイルのOPEN CLOSEを繰り返すことができる。こうしておくと、作業は人が付いていなくても実行できる。また、パソコン上で稼働する各種のソフトウエアを利用できることも強みとなる。各種のスクリーンエディタの利用をはじめ、検索されたコピーファイルとプログラムの関係や日付項目名のデータ型とそれを含むプログラムとの関係などを整理し、検索できる簡易なシステムをパソコンで稼働するデータベースソフトを利用して作ることも容易である。少なくともプログラムの棚卸し段階、日付項目が検索されたあとの検討段階にはこうした検索手段があるかないかが、作業の進捗に大きく影響しよう。さらに、汎用機種やオフコンでは JCLやプロシーデュアの作成など操作のための準備に面倒が多く、未熟練者を補助者として同時並行的に作業を進めることには困難がともなう。その点、操作の容易なパソコンでは複数の未熟練者を作業に動員することが可能になる。

パソコンにダウンロードして作業を行ったときの問題点として懸念されていたのは、プログラム中の漢字定数やファイル中の漢字データを戻したときにシフト IN/OUT 符号の関係から字化けを起こさないかという点と、ファイルの持つ標準レーベルであった。

現時点でわかっていることは、ほとんどの汎用機種の通信サーバに、端末として接続されているパソコンとファイル転送ルーチンを通じて漢字データのやりとり、つまりアップロードしても字化けを起こさないような機能を持たせていることである。標準レーベルの問題も転送ルーチンでは解消されると考えてよさそうである。

いまひとつ、汎用ユーザに多いのはパソコンの性能に関する不安である。その点を明らかにするため、今回開発したツールについて実行時間の計測を行った。不満があればパソコンを複数台使うことでスループットの向上をはかればよい。現用機種の増設に比較すればはるかに少ない投資ですむ。本学では数十台の並行処理は可能であろう。計測に使用したプログラムはもっとも実行時間の長い日付項目の文字列検索と入出力時間がネックとなるJCL 解析用のプログラムである。

このプログラムは文字列の部分検索を行うため内部処理の計算量が多いことが予想されていたもので、実測結果もそれを裏付けるものであった。測定方法は、簡単に対象とするファイルのOPEN/CLOSE直後のシステム時間を画面に表示させ、それによって時間を計算したもので、正確ではないがおよその処理時間は推定できる。

プログラムの行数、バイト数、二次検索された語数と実行にかかった時間は次のとおりであった。１次検索の部分文字列は１５個である。プロセスネックの文字列処理だけに内部速度の速いパソコンに切り換えると、効果的に改善できることも判明した。

JCL の解析のほうはJCL ファイル10本、 492行、 23314バイト分の処理にかかる時間である。 25Mhz程度のマシンでも本学にある JCLファイル約1100本の処理に実質20分くらいの所要時間と推定できる。

JCL解析の実行時間		PC9821 25Mhz		
処理対象 10本	FDD	HDD	RAMﾃﾞｨｽｸ	
492行 23314バイト	5 秒	1 秒	0.5 秒	
文字列検索	PC9821 CE2		FMV 5DH	

の実行時間				25Mhz		133Mhz
処理対象						
	行数	バイト数		FDD	RAMﾃﾞｨｽｸ	FDD
				分 秒	分 秒	分 秒
B100	393	15976	+8	1:58	1:53	0:17
B270	916	39819	+11	4:11	4:02	0:56
B290	417	16659	+10	1:57	1:53	0:42
B300	240	10783	+3	0:47	0:44	0:06

+ は二次検索の語数

5　ツール開発にあたっての留意事項と制限事項

制限事項

本ツールは、本学内で使用できることを優先条件とし、大規模なリアルタイムトランザクション処理への適用やデータベースのデータ定義の調査などには触れないことにした。制限事項については各ツールにより登録できる文字列数、ルーチンの数などを設けたが、桁数引き伸ばしのさいのファイルにあるレコードを扱うときは共通制限事項として論理レコード長4000字以内、レコード記述中にある OCCURS 配列は１次元のみ対応、マルチレイアウトレコードには対応しないこととした。

留意事項

ツールの開発にあたって特に留意したのは操作性とカストマイズの容易性である。これは、操作の主力に学生バイトを想定したこと、日付項目の検索に用いる文字列の追加や置換したい標準ルーチンの追加などの頻度がかなり高いと予想されたからである。

ａ．連続実行

自動的に連続実行させることで可能な限りの省力化を図ることと同時に、調査漏れになるプログラムの発生を防ぐことができる。本学では、サブシステム（業務）単位にチェック対象とするプログラム名を順に記録し、それに業務名をつけたファイルを作ってサブシステム単位での連続実行を予定している。ただし、サンプル調査のための利用も考慮して、１本ずつの実行も可能としてある。

登録ファイルの例

```
GMGK       <--- ファイル名
GMGKYO10         ------------
GMGKYO20         ------------
GMGKYO30         ------------
```

このファイルには、文字列の検索 COPY 句の抽出などを処理した日付を記録し、調査資料の作成済みを確認できる。棚卸しや調査段階ではとくに漏れの発生を防ぎたいため本学ではＪＣＬからプログラム名を抽出して登録ファイルを作成し、その順に処理したプログラムからコピーファイル名の登録ファイルを自動的に作成する。

なお、登録したプログラム名の誤りのため実行が中断しないよう、エラーログをとるファイルに記録を残し次のファイルの処理に進む機能も組み込んだ。

汎用機種やオフコンと違って、パソコンでのプログラム言語では、C 言語でも COBOLでも、これからOPENするファイルを、そのファイルの存在する装置・ディレクトリ・ファイル名・拡張子を変数として

標準入出力ルーチンに引き渡すことにより、動的に指定することができる。キーボードから入力した変数でも、プログラム内部の定数でもよいし別のファイルから入力したデータでもよい。このような処理は、かってのＴＳＳをベースとするOSでは可能であったが、現在使用されている汎用機種やオフコンではJCL で静的にファイルを割り当てる関係から、かりに再使用可能プログラムを作成しておいても、簡単に次の入力ファイルを指定することが難しい。そのうえ、プログラム領域にある原文ファイルをデータとして使用することが許されない機種もあった。オペレータの介在しないＴＳＳでは端末から送るメッセージをファイルに格納したあとそれをプログラムとして扱うかデータとして処理するかは使用者の自由であった。

また、パソコンのMS/DOS標準形式である行順ファイルは、レコードごとの区切りを復帰・改行符号（CR,LF）で示しているので、4000バイトと指定したバッファエリアにそれ以下の長さのレコードであれば、問題なく１レコードずつ呼び出せるという特性がある。汎用機種で主流を占めるブロック化した固定長レコードのように、プログラム中のレコード記述に実際に扱うデータの論理レコード長を書いておかなくても処理は正しく行える。

ｂ．操作方法の統一

ツールごとの実行時の操作はできるかぎり統一して、パソコン操作時の混乱をふせぐようにした。スタート時のメッセージはほとんどすべてのツールで次のようになる。

データを入力するドライブは？(A/B/C/D ..)

プログラム名を登録したファイルを使いますか?(Y OR N)

業務別にチェック対象のプログラムを登録しているときは

Y と打つと　　続けて

プログラムを登録したファイル名?　の入力を求める

N としたときは

プログラム原文のファイル名?　の入力を求める

ｃ．カストマイズの容易性

日付項目の検索のさいには前述どおりどのような名称が用いられているかわからないので、サンプルをいくつか検索したうえで検索に使う部分文字列を指定できるように考慮しておく必要がある。このツールは、既定値として COPY CALL ENTER DATE などをプログラム定数として持ち、YEAR NEN YY HIZ TUKなどの文字列をパラメータファイル に持たせて、ユーザがそれらを取捨選択できることと、実行時に追加した文字列をこのファイルに追加していけるようにしてある。

また、プログラム内部対応のためには標準的な手続きとの置き換えを行う方法をとるが、日付項目に関する演算や処理方法についてはユーザごとの特殊事情による多様化が避けられない。それらを考慮して標準ルーチンを多数用意するとかえってその選択を誤りやすいので、標準ルーチンをユーザ自身が簡単に作って追加できるように配慮することとした。

例　経過年数の計算

```
P1100  03
P1101Y     MOVE  ?1        TO   WWY4NEN1.
P1102      IF    WWY4NEN1  <    WWKIJYUN  ADD  100  TO  WWY4NEN1.
```

```
P1103Y     MOVE  ?2        TO   WWY4NEN2.
P1104      IF    WWY4NEN2  <    WWKIJYUN  ADD  100  TO  WWY4NEN2.
P1105Y     COMPUTE   ?3  =  WWY4NEN1  -  WWY4NEN2.
```

d．現在使用中の正しいデータの再利用による誤転記防止

現在の処理に使用されている正しい素材は、できるだけ活用することとした。たとえば、プログラム中のデータの記述をファイル形式変換の指示用のデータに使用するというようなことである。人手による誤りを可能な限り防ぐことが目的であるが同時に、省力化も図れる。コピー句単位でレコード形式を記述する場合は、そのファイルを利用して桁数を増加する項目、減少させる項目などの指示データを作成できる。

6　開発環境と使用言語

ツールの開発には次のようなパソコンとコンパイラを使用した。

機種	コンパイラ	ＯＳ
FMV DESK POWER	MICRO FOCUS COBOL/2	WINDOWS95
PC9821 CE2	MICRO FOCUS COBOL/2	MS/DOS 6.2

開発言語として COBOLを使用したのは汎用機種やオフコンへの移植可能性を考慮したためでもある。パソコン固有の機能を使用する部分は特定のモジュールにまとめ移植のときは取り外せるようにしてある。なお、これらツールの開発はCOBOL でなくC 言語のほうが適していると考える人が多いようであるが、現行規格のＣ言語を使用すると、順ファイル・行順ファイルいずれの入力にも使用できる関数をはじめ、COBOL の STRING UNSTRING INSPECTなどの命令に対応する多機能関数を新たな関数として作成する必要があり、とり急ぎ開発しておきたいツール群はCOBOL で開発することにした。

7　おわりに

本学での対応作業は５月に始める予定を立てている。学生に対するホストの操作の指導、パソコンで使用するツール群の機能の説明などに１月ほどの時間をかけ、その間にホスト側の環境整備やドキュメント類の整理などを行う。６月から本格的な稼働に入るが、準備は順調である。結果として、パソコンの利用によりホストの負荷を軽減すること、複数台のパソコンによる並行作業でスケジュールの調整が楽にできること、作業の単純化によりバイトの利用を可能にすることなどは実現できそうである。

参考文献

1)情報システムの西暦2000年対応の実務資料集:日本情報システムユーザ協会:1996.6

2)IBM 西暦2000年対応計画・導入ガイド第４版:IBM:1996.9

3)大林久人・越智洋之:2000年問題の傾向と対策:エーアイ出版:1997.3