Ｕｎｉｘファイル変換ツールの開発

5L-1

鷲澤正英　　菅野文友
（東京理科大学　工学研究科）

１．はじめに

情報通信は，情報処理機能の分散化，ネットワークによる統合化，および音声・データなどのメディア統合化の方向へ急速に進展している．さらに，対象とする計算機とは異なる別の計算機を使用した，クロス開発環境下でのソフトウェア開発が多くなってきている．このような傾向に対処していくため，設計，製造から試験までの作業を効率化（ソフトウェア開発環境の整備など）し，ソフトウェアの生産性を高める必要がある．ソフトウェア開発環境の構築には，まず有用なソフトウェア・ツールの設定が，必要不可欠である．

ＯＳの異なる計算機間で，クロス環境の整備によって，ソフトウェアを再利用することは，既作成ソフトウェアの有効活用の意味でも，有益である．そこで，Ｕｎｉｘ上で開発したソフトウェアを他の環境でも再利用可能にするファイル変換のツールを，部品化を考慮して作成した．

２．Ｕｎｉｘファイル探索アルゴリズム

Ｕｎｉｘのファイルシステムは，ツリー状のディレクトリ構造をなしており，再帰的手続きを行なうには最も適していると考えられる．しかし，再帰手続きには，コール／リターンを何重にもネストしていくため，スタックの浪費と処理スピードの低下をもたらす可能性があるという欠点がある．しかし，このことによる影響は，微々たるもので，再帰解によって明解なアルゴリズムが得られるならば，これを選ぶべきである，と考える．本ツールでは，ツリー状ディレクトリの探索，および指定したファイルのサイズが非常に大きくなった際，格納ブロックが１０ブロックを超えた場合の，間接方式によるブロック探索で用いた．

概略を，図１に示す．

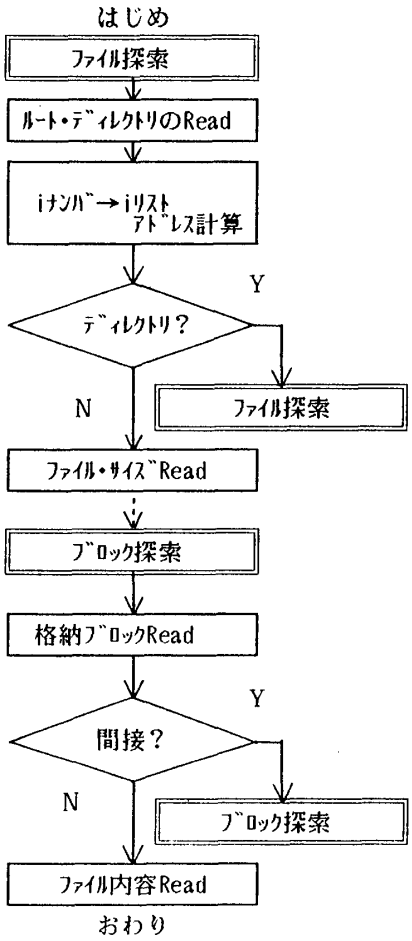

図１　ブロック探索アルゴリズム

３．プログラムの部品化

プログラムの構成要素を標準化し，再利用可能にするものとして，部品化がある．本ツールでは，図２に示すように，Ｕｎｉｘファイル入出力の関数を部品化を考慮して作成した．これは，ＭＳ－ＤＯＳの環境下で，Ｕｎｉｘファイルをアクセスする場合に利用可能であるが，Ｕｎｉｘファイルの論理ブロックの対応が，各社様々であるため，汎用性はないものになっている．

On the conversion tool of some Unix files
Masahide WASHIZAWA , Ayatomo KANNO
Faculty of Engineering , Science University of Tokyo

```
・  fp  = ux_open( filename,flag );
        char *filename; 指定ﾌｧｲﾙ名
        char *flag;     ｵｰﾌﾟﾝ･ﾓｰﾄﾞ
        int  fp;        ﾌｧｲﾙ番号
・  ret = ux_close( fp );
        int  ret;       結果
        int  fp;        ﾌｧｲﾙ番号
・  ret = ux_read( fp,buff,length  );
        int  ret;       結果
        int  fp;        ﾌｧｲﾙ番号
        char *buff;     入力ﾊﾞｯﾌｧ
        int  length;    入力文字数
・  ret = ux_write( fp,buff,length );
        int  ret;       結果
        int  fp;        ﾌｧｲﾙ番号
        char *buff;     出力ﾊﾞｯﾌｧ
        int length;     出力ﾃﾞｰﾀﾊﾞｲﾄ数
```

図２　Ｕｎｉｘ・Ｉ／Ｏ関数

４．ツールの概要

本ツールは，ＵｎｉｘファイルとＭＳ－ＤＯＳファイルのファイル変換を行なうツールである．また，本ツールは，図３に示す，特定のＵｎｉｘおよびＭＳ－ＤＯＳの仕様を前提としており，実行は，ＭＳ－ＤＯＳ側で行なうものとした．さらに，開発用言語は，ディスク・アクセス部を８０８６アセンブラで，他をＣ言語で記述した．

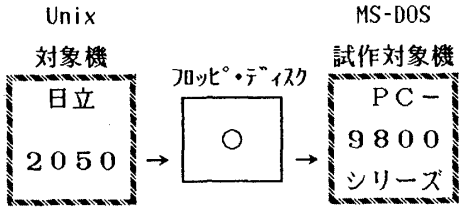

図３　ツール概略図

本ツールの機能は，

①ＵｎｉｘファイルからＭＳ－ＤＯＳファイルへのコンバート　（作成日付，モードを含む）

②ＭＳ－ＤＯＳファイルからＵｎｉｘファイルへのコンバート　（作成日付を含む）

③ＵｎｉｘおよびＭＳ－ＤＯＳファイルダンプがある．システムとしては，エラーリカバリ機能を充実し，さらにポップアップメニューによって，操作性を向上させた．

５．おわりに

Ｕｎｉｘの仕様は，メーカ毎に異なっているため，本ツールでは，汎用性に問題が残る．以下に，今後の課題，考察を示す．

①汎用性

Ｕｎｉｘの標準化を巡る論議は，ＩＥＥＥのＰｏｓｉｘ規格設定を中心に，益々活気を帯びた様相を呈している．今回のツールは，特定の仕様に限定して作成したが，今後は，標準化動向を見守りながら，汎用性も考えて行きたい．

②部品化技術適用への考察

部品化を行なう場合には，部品となりうるモジュール部分の独立性が保たれなければならない．本ツールでは，Ｕｎｉｘファイルの入出力を部品化の対象として作成したが，モジュールの独立性に関して，保持できたものと考えている．Ｃ言語では，モジュールの構成単位が，関数なので，独立性の低下は，他の言語に比較すると，少ないものと考えられる．しかし，外部変数の存在や，ポインタ変数の使用が不可避であるため，関数の呼び出し関係がモジュールの中に残る点など，モジュールの独立性を損なう部分もある．また，本来プログラム作成を助ける機能である“#include”や“#define” なども，モジュールの独立性の観点からは，好ましくないものと考えられる．

以上

【参考／引用文献】

[１] 片岡雅憲：「ソフトウェア・モデリング」，日科技連，（1988）．

[２] Marc J.Rochkind：「Advanced Unix Programming」，アスキー出版，（1987）．

[３] 高橋裕之：“ソフトウェア部品化技術”，情報処理学会第３５回全国大会，pp.945-946，（1987）．