UTSのシェルとファイルシステム

松延, 栄治
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1468155

出版情報：九州大学大型計算機センター広報. 20 (5), pp.407-423, 1987-09-25. 九州大学大型計算機センター
バージョン：
権利関係：

UTSのシェルとファイルシステム

松延栄治*

1. はじめに

UNIXオペレーティングシステムは3つの基本構成部分，すなわちスケジューラ(scheduler)，ファイルシステム(file system)，シェル(shell)からなっている．本稿ではこのうちファイルシステムとシェルについて述べる．最後に，通信用コマンドについて簡単に紹介する．

2. ファイルシステム

本節ではディレクトリ構造，ファイル構造，ファイル名とファイル操作，ファイルのアクセス権について説明する．ファイルの内部構造については利用者はほとんど意識する必要がないので省略する．

2.1 木構造の階層ディレクトリ

UNIXのファイルシステムにはファイルとディレクトリがあり，木構造の階層ディレクトリをしている．利用者はディレクトリ構造だけを知っていればよく，ファイルシステムがディスク上にどのように構成されているかは意識する必要がない．本センターのUTSのディレクトリ構造を図1に示す．"/"はルート(root)と呼ばれる．図中の"葉"(leaf)にあたるのが通常のファイル(例:test.c)でそれ以外はディレクトリである．ディレクトリは管理簿(名簿)に相当する．

対象となるディレクトリあるいはファイルにアクセスする場合には，自分が現在どこのディレクトリにいるのかが問題となる．利用者は必ずファイルシステム中のどこかのディレクトリに位置づけられている．現在位置づけられているディレクトリをワーキングディレクトリ(カレントディレクトリ)という．先の例でワーキングディレクトリがa70001aであるとき図中のtest.c を指定する方法は次の2つがある．

(1) /usr/usr1/a70001a/test.c
(2) test.c

(1)はルートから目的のファイルに至るパス上のディレクトリ名を"/"で区切って並べ，最後に目的のファイル名を書くことにより指定する方法である(完全パス名での記述)．

昭和62年8月3日受理

*九州大学大型計算機センター研究開発部

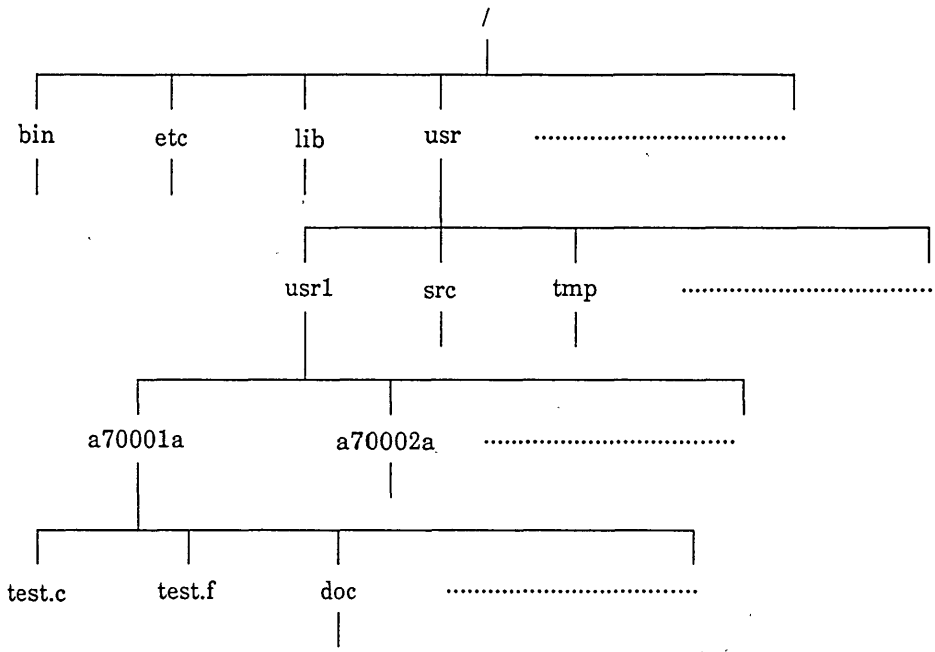

図1　UTSのディレクトリ構造

(2)はワーキングディレクトリから目的のファイルに至るパス上のディレクトリ名を"/"で区切って並べ，最後に目的のファイル名を書くことにより指定する方法である(相対パス名での記述)．コマンドの引数にパス名(ファイル名)が使われる時はどちらの指定方法を用いてもまったくかまわない．

UNIXではログイン時に各利用者固有のディレクトリに位置づけられている．これを利用者のホームディレクトリという．本センターのUTSでは利用者a70001aのホームディレクトリは/usr/usr1/a70001aに設定している．現在の自分のワーキングディレクトリを知るにはpwdコマンドを用いる．投入したコマンドの前にある%はシェルのプロンプトである．

```
% pwd
/usr/usr1/a70001a
```

ワーキングディレクトリの内容を見るにはlsコマンドを用いる．このコマンドはMSPのLISTCコマンドに近い働きをする．

```
% ls
doc
test.c
```

```
:
:
```

もし他のディレクトリの内容を見たければ, 次のよう入力する.

```
% ls ディレクトリ名
```

内容の詳細な情報を知りたければ, lsコマンドに−lオプションを付ける. また, ファイル名の先頭が .(ドット)であるファイル(例えば,.,..,.login,.login)まで表示させたいときは, 更に−aオプションを付ける.

```
% ls -a -l
total 30
drwxr-xr-x  27 a70001a  usr1         496 Jul 31 15:40 .
drwxrwxrwx  28 usr1     usr          496 May 26 12:02 ..
-rw-r--r--   1 a70001a  usr1         455 Feb 27 19:18 .cshrc
-rw-r--r--   1 a70001a  usr1        1044 Feb 27 19:18 .login
dwxr-xr-x    2 a70001a  usr1          80 Jul 31 15:40 doc
-rw-r--r--   1 a70001a  usr1         287 Jul 10 19:34 test.c
-rw-r--r--   1 a70001a  usr1        2984 Jul 10 18:18 test.f
 :
 :
```

lsコマンドで出力された結果の各欄は次のことを意味している. 最初のtotalは4096バイトを1ブロックとしたときのブロック数を示している. 出力された左端の文字でdはディレクトリであることを示し, −(ハイフン)は一般のファイルであることを示している. 次の9文字はファイルに対するアクセス権を示している(詳細についてはファイルのアクセス権の節2.4を参照). 次の数字は枝(リンク)の数である. 更に順に所有者名, グループ名, サイズ(バイト単位), 最新更新日時, ファイル名が並んでいる.

ここでコマンドについて簡単に触れておく. コマンドの入力が複数行にまたがるときは行末に"\"を入力してコマンドを継続することができる. コマンドはオプション指定には"−"を付けて表す. オプションを複数指定するときは1つのハイフンの後にまとめて置くことができる. 文字の間を空白で区切る必要はない. またオプションの文字列は順不同である. すなわち, いまのコマンドは

```
% ls -la
```

としても同じ結果が得られる.

さてlsの出力で, ".", ".."で参照されているファイルがあるが, "."はワーキングディレクトリ自身を表し".."はワーキングディレクトリの親ディレクトリを表している. これらの記号はパス名として使うことができる. 例えば,

```
% ls ..
```

とすれば，ワーキングディレクトリの親のディレクトリの内容を見ることができる．

ワーキングディレクトリの移動にはcdコマンドを用いる．cdコマンドの使用法は次の2つがある．

(1) cd　パス名

(2) cd

引数なしで実行すると利用者のホームディレクトリに無条件に戻る．ワーキングディレクトリを希望するディレクトリに移したいときにはパス名を与える．次のように .，..を用いたパス名の指定も当然可能である．

```
% pwd
/usr/usr1/a70001a
% cd ../../src
% pwd
/usr/src
```

新規にディレクトリを作るにはmkdirコマンドを用いる．mkdirの引数が新しいディレクトリのパス名になる．ワーキングディレクトリの移動はないので，新しいディレクトリでの作業を行うにはcdコマンドでワーキングディレクトリを変更する必要がある．

```
% pwd
/usr/usr1/a70001a
% mkdir letter
% pwd
/usr/usr1/a70001a
% ls
letter
doc
test.c
:
:
```

ディレクトリの削除はrmdirコマンドを用いて行う．

```
% rmdir doc
```

この場合docにはディレクトリ，およびファイルが登録されていてはいけない．もし登録されているときはdoc内の全てのディレクトリ，ファイルを削除してからでないとdoc自身は削除できない．ファイルの削除についてはrmコマンドを用いる(3.2参照)．

2.2　ファイルの構造

UNIXにおけるファイルは単なる「バイト列」であり，MSPのようにファイル編成，レコード長，ブロック長といった概念はない．このため，MSPでの新規ファイルの作成時のような面倒な手続きはいらない．UNIXではファイル領域が必要となった時点で，システムによりブロック単位(UTSでは1ブロック4096バイト)で確保される．あらかじめ領域を確保する必要はない．ここでいうブロックとはファイルごとに異なるものではない．ディスクの全領域がディスクアクセスの単位である固定長のブロックに分割されており，このブロックを意味している．普通，利用者はこのことを意識する必要はない．なお本センターのUTSでは1ファイルの領域の上限は4メガバイトで運用している．ファイルは次の3つのタイプに分類できる．

(1)通常ファイル
(2)ディレクトリファイル
(3)特殊ファイル

通常ファイルは利用者が一般に扱うファイルである．ディレクトリもUNIXではファイルとみなしている．特殊ファイルとは入出力のドライバルーチンとのインタフェイスをとるためのファイルである．磁気ディスク，磁気テープ，プリンタ装置などが特殊ファイルに対応している．これらの入出力装置に対する入出力処理は対応する特殊ファイルへの入出力処理を行うことにより実現できる．

2.3 ファイル名とファイル操作

UTSではファイル命名には次の注意が必要である．

(1)大文字と小文字は区別される．
(2)最大14文字までの任意の文字列である．
(3)次の特殊文字は使用できない．

; | 〉〈 & * [ ] ? ”

一般にファイル名は利用者が任意につけてよい．しかし，ファイル名の先頭が.(ドット)で始まるファイルはシステムで使用するファイル(.login, .cshrcなど)になっているので便宜上，先頭にはあまり.(ドット)をつけないほうがよい．言語プロセッサが処理するファイルは特定の形式をしている必要がある．ファイルの最後の2文字が“.”プラス“特定文字”となる．次に種々のファイルタイプに対する特定文字を示す．

C言語ソースプログラム	***.c
Cヘッダファイル	***.h
Fortran77ソースプログラム	***.f
アセンブラソースプログラム	***.s
Pascalソースプログラム	***.p

オブジェクトプログラム	***.o
Yaccソースプログラム	***.y
Ratforソースプログラム	***.r
Lexソースプログラム	***.l
Eflソースプログラム	***.e

ファイル内容を表示させるものとして，最も基本的なものはcatコマンドがある．catの名はconcatenate(連結)から取られていて，もともと複数のファイルを連結する働きがある(3.1参照)．

```
% cat file1 file2
```

とすればfile1とfile2を連結して表示させることができる．

```
% cat file1
```

としたときはcat本来の意味はなくなり，ただ単にfile1の内容を画面に表示させるだけである．長いファイルをcatで表示するのは面倒である．tailコマンドはファイルの最後の数行，または途中の一部分を表示するのに便利である．

例えば，最後の5行を表示したければ，

```
% tail -5 file1
```

とすればよい．もし最初の何行かをスキップさせたいときは－ではなく＋を用いる．最初の4行を読み飛ばして，5行目から表示させたければ，

```
% tail +4 file1
```

とする．

ところでコマンドの操作でファイルを指定する場合，ある特定のパターンに合致するファイルをすべて指定したいことがある．UNIXではこのようなパターン整合機能を次のような特殊文字(ワイルドカード文字ということもある)を用いて実現している．

(1)	*	空文字列を含む任意の文字列と一致する
(2)	?	任意の1文字と一致する
(3)	[ ]	[ ]で囲まれる任意の1文字と一致する "－"で範囲を指定した場合はその間の任意の1文字と一致する

例えば，

```
% cat a*.c
```

とするとaで始まるcソースプログラムすべてのファイルの内容を表示する．また，a[1－3].cはa1.c, a2.c, a3.cというファイル名を生成する．a[235].cとすると，a2.c, a3.c, a5.cを生成する．*, ?, [ ]を組み合わせて使うこともできる．?abc*は2文字目から4文字目が“abc”となる任意の長さのファイル名を生成する．

次にファイル操作コマンドをいくつか紹介する．ファイルをコピーするにはcpコマンドを用いる．例えば，/usr/usr1/a70002a/ex1.cを利用者a70001aのホームディレクトリへコピーしたければ，

```
% cp /usr/usr1/a70002a/ex1.c  ex1.c
```

とすればよい．ただし，このときのワーキングディレクトリはホームディレクトリであるとする．同じ名前であれば，次のように入力してもよい．

```
% cp /usr/usr1/a70002a/ex1.c  .
```

“.”はワーキングディレクトリを示す記号である．コピー先のディレクトリに同じファイル名がなければ自動的に作成される．もしあれば，上書きされ，元の内容は失われる．今の例で異なるファイル名,例えば，ex2.cにコピーしたければ，

```
% cp /usr/usr1/a70002a/ex1.c  ex2.c
```

とすればよい．ファイルの名前を変えたり，移動したりするにはmvコマンドが使われる．cpコマンドとの違いは移動元のファイルは残らないことである．

```
% mv test.c ex1.c
```

このような同一ディレクトリ内のファイルの移動は名前の変更として機能する．

```
% mv *.c /usr/usr1/a70001a/src
```

とするとcのソースプログラムを別のディレクトリの下に一括して移動することができる．ファイルの削除にはrmコマンドを用いる．例えば，

```
% rm *.o
```

とすれば，ワーキングディレクトリにあるオブジェクトプログラムのファイルだけをすべて削除する．−iオプションをつければ，rmの実行時に削除してよいか聞いてくる．yを入力すれば削除され，nを入力すれば，削除されない．複数のファイルを会話的に選別して，削除したいときは便利である．

最後に検索に関するコマンドについて述べる．パターン整合機能をechoコマンドで用いて，ワーキングディレクトリ内でパターンに一致するファイル名を検索表示することができる．echoコマンドはもともと「echo 引数」の形式で用いて，ただ単に引数を画面に表示するだけである．しかし，例えば，

```
% echo ????
```

とすることによりワーキングディレクトリ内の4文字からなるファイル名を検索表示できる．また，

```
% echo ? abc*
```

とすれば1文字からなるファイル名とファイル名の先頭がabcで始まるファイル名を検索表示する．

あるディレクトリより下にあるファイルの検索にはfindコマンドを使用し，次の形式で用いる．

% find　パス名　オプション

パス名で指定したディレクトリよりも下にあるすべてのファイルに対してオプションで示す条件に合致するファイルを検索する．オプションには次のようなものがある．

-name file名	file名で指定したファイル名と一致したとき真
-size n	ファイルサイズがnブロック長のとき真
-atime n	ファイルがn日以内にアクセスされているとき真
-mtime n	ファイルがn日以内に変更されているときに真
-ctime n	ファイルがn日以内に作成されているとき真
-print	常に真で他のオプションと組み合わせて用いられ，他のオプションの条件に合致するファイルのパス名を出力

オプションは複数組み合わせて使用できる．その場合，各項を並べただけのときは各項の論理積を取り，−oで組み合わされたときは論理和，オプションの前に!を指定したときにはそのオプションの条件が成立しないときに真であることを意味する．

%find / -name test.c -print

とすると全てのファイルの中からtest.cという名のファイルを捜して，そのパス名を表示させることができる．

ファイルの中から特定のパターンを検索するのに便利なのがgrepコマンドである．grepコマンドは単純な文字列の検索から正規表現(Regular Expression)による複雑なパターンの検索まで，幅広く使える．コマンドは次の形式をしている．

grep オプション パターン file1, file2,・・・・・

パターンは次のような特殊記号を用いて正規表現を表す．

.	任意の1文字
*	直前の文字の任意個の繰り返し
^	行の先頭
$	行の最後
[]	カッコで囲まれた文字が検索される文字の集合を示す
*	任意の文字列

オプション

−v	パターンが一致した行以外の全ての行を出力
−c	パターンが一致した行数だけを出力
−n	一致したパターンを含む行のファイルのなかで相対番号をつけて行の内容出力
−l	パターンが一致した行を含むファイル名出力

-y　　パターンに含まれる小文字は大文字とも一致する

例えば，行の先頭が数字で始まる行をex.textから検索して，条件に一致する行数を出力するには，

```
% grep -c '^[0-9]' ex.text
```

とすればよい．パターンが特殊文字や空白を含むときは'(アポストロフィ)で囲む必要がある．またaで始まりedで終る文字列を検索するには，

```
% grep 'a.*ed' ex.text
```

とする．

2.4 ファイルの所有権と保護

すべてのファイルには必ずそのファイルの所有者(owner)がいる．所有者とは基本的にはファイルやディレクトリを最初に作成した人である．またおのおの利用者は1つの利用者グループに属している．1つのファイルにおいて，所有者，グループ，一般利用者別々に，読み出し権(read)，書き込み権(write)，実行権(execute)のそれぞれのアクセス権を設定できる．次にそれぞれのアクセス権の意味を示す．

	ファイル	ディレクトリ
読み出し権	内容を見ることができる	ディレクトリ内の名前を見ることができる
書き込み権	内容を変更できる	新規にファイルを作成したり，ファイルを削除できる
実行権	ファイルをコマンドとして実行できる	ディレクトリ内の詳細情報を知ることができる

ファイルの削除にはそのファイルの書き込み権があるだけではだめで，そのファイルが存在するディレクトリの書き込み権が与えられていなければならない．

ファイルのアクセス権に関する情報はlsコマンド(-lオプションつき)で知ることができる．

```
% ls -l kanji.c
-rw-r--r--   1 a70001a    usr1     19681 Jul 10 19:30 kanji.c
```

2文字目から9文字を3文字ずつに区切って，左から，所有者，同一グループ，一般利用者に対してそれぞれ読み出し(r)，書き込み(w)，実行(x)を許可するかどうかのマークがしてある．r, w, xがマークしてあれば，各々の操作ができることを意味している．このkanji.cというファイルは所有者だけが読み出し，書き込みが許されており，その他は読み出ししか許さ

れていない．アクセス権を変更するにはchmodコマンドを用いる．上の例で一般利用者の読み出し権を取り消すには

```
% chmod  o-r  kanji.c
```

とすればよい．oは一般利用者を意味し，他にu(所有者), g(グループ), a(全員)を指定できる．－は取り消しを意味し，＋は追加を意味する．rは読み出し許可を意味し，他にw, xが指定できる．MSPのように特定の利用者へのアクセス権の設定はできない．

3. シェル

端末利用者が入力したコマンドはシェルと呼ばれるコマンドアナライザで解釈される．入力コマンドが，実行可能であればシェルによりそのコマンドのプロセスが生成される．コマンドのプロセスが実行権を得ると，カーネルの機能を使用しながら端末とのやりとりをシェルを介さずに直接行ったりして，処理する．コマンドの処理が終了すると端末の制御はシェルに戻る．こうして，シェルは再び端末からの入力を待つことになる．このように端末利用者が投入したコマンドを処理するプログラムすなわち，コマンドアナライザのことをシェルと呼ぶ．このシェルはコマンドを解釈しながら実行する一種のインタプリタと見なすことができる．現在，本センターのUTSで使用可能なシェルは次の4つがある．本センターのUTSではログイン時c shell(以下cshと略す)が起動されるようにしており，本稿の説明ではcshを用いている．

(1) sh　　ベル研生まれの標準的なシェルである．設計者の名をとってBourne shellと呼ばれることもある．

(2) rsh　　使用できるコマンド等に制限を受けている限定版のシェルである．rshのrはrestrictedの意味である．

(3) jsh　　コマンドレベルで日本語を扱えるようにしたshの拡張版である．

(4) csh　　バークレー版のシェルで，研究開発向きである．使いなれた人にとっては便利な機能を多く持っている．

3.1 標準入出力とリダイレクション(redirection)

UNIXでは標準ファイルと呼ばれる論理的な入出力装置名により入出力処理を行っている．標準ファイルには次の3つがある．

(1) 標準入力
(2) 標準出力
(3) 標準エラー出力

通常，標準入力は端末のキーボード，標準出力，標準エラー出力は端末画面に割り当てられている．標準ファイルはシェルによりオープンされるから利用者がわざわざオープンする

必要はない．入力をキーボードからではなく，ディスクファイルから行う時は標準入出力装置を切り換える必要がある．これをリダイレクションという．切り換えには次の3つの種類がある．

(1) 入力をファイルから読む．
(2) 出力をファイルに書き出す．
(3) 出力を既存のファイルに追加書きする．

これらの指定にはそれぞれ<, >, >>という記号を用いる．

例えば，catコマンドで

```
% cat file1 file2
```

とすればfile1とfile2を連結して標準出力ファイルに出力されるだけであるが，これを次にようにすると，2つのファイルが連結されてnewfileという名前のファイルに出力される．

```
% cat file1 file2 >newfile
```

入出力ともリダイレクションを用いてもかまわない．システムコマンド，例えばccコマンド(ccコマンドは文献[5]参照)で，

```
% cc <test.c
```

のように<を使ってリダイレクションを行ってもよい．しかし，多くのシステムコマンドではファイル名を引数に取るので単に次のようにしてかまわない．

```
% cc test.c
```

3.2 パイプライン(pipeline)

複数のプログラムを順次実行させるには次のように，コマンドを";"を使って区切って並べればよい．

```
% command1 ; command2  ; command3
```

しかし，複数のプログラムを連続して実行する場合に，あるプログラムの標準出力を次のプログラムの標準入力に結合したい場合にはパイプラインを用いればよい．パイプラインの記号は"|"で表す．

```
% command1 |command2 .....
```

これは次のようにコマンドを実行したのと同じである．

```
% command1 >temp1
% command2 <temp1 >temp2
:
:
```

但し，パイプラインではファイルtemp1, temp2, …は使わない．次にパイプラインの例を示す．

```
% who |wc -l
```

この例は現在のログイン名のリスト出力ファイルをコマンドwcの入力に引き渡す．wcはファイルの行数をカウントして出力する．すなわち，現在のログインユーザ数が出力される．

```
% who |grep 'a7.*[a-j]' |sort
```

この例は現在のログイン名から特定の条件に合致するものだけを拾い出し，ソートして表示する．

一般に標準入力を加工して，標準出力へ結果を出力するプログラムを“フィルタ”という．UTSではいろいろなフィルタ(cat, ed, grep, pr, sh, sort, wcなど)が用意されている．

3.3　プロセスとバックグラウンド処理

UNIXでは，各々独立した実行単位をプロセスと呼ぶ．プログラムの実行がなされるとプロセスが起動され，プログラムの実行が終了するとプロセスは消滅する．1つのプログラムの実行中には1つのプロセスだけが生成されるとは限らず，2つ以上のプロセスが生成されることもある．プロセスは他のプロセスを起動させたり，消滅させたり，更に他のプロセスと通信することができる．シェルも1つのプロセスとして動作する．実行されるプロセスにはシステムでユニークなプロセスID(PID)が付けられる．利用者に関連したすべてプロセスはpsコマンドを入力することにより確認できる．

```
% ps
   PID     TTY  TIME COMMAND
   736  tty084  0:00 csh
   889  tty084  0:00 csh
   891  tty084  0:00 ps
```

特定のプロセスをkillコマンドを用いて強制的に終了させることができる．その場合，そのプロセスのPIDを知っておく必要がある．

端末との会話的な処理(フォアグラウンド処理)とは別に，実際の終了を待たずにバックグラウンドで行う処理をバックグラウンド処理という．MSPのFIBジョブに相当するがJCL相当のものは必要ない．UTSではただ単にコマンドの最後に“&”を付けるだけでよい．

```
% cc test.c &
101
%
```

この例ではccコマンドをバックグラウンドで処理している．このプロセスのプロセスIDは101であることが表示されている．

3.4　シェルプロシジャ(シェルスクリプト)

シェルはコマンド・インタプリタであると同時に高級なプログラミング言語である．コマンドの組み合せからなる実行可能なファイルをUTSではシェルプロシジャ(シェルスクリプト)と呼ぶ．MSPでのコマンドプロシジャに相当する．シェルプロシジャではシェル変数と呼ばれる変数を用いることができる．このシェル変数には利用者が定義して使用する利用者定義変数，システムであらかじめ定義されている特殊変数及び環境変数がある．また，コマンドの実行の流れを変更する制御機能，引数の受け渡し，シェルプロシジャ内の内部コマンドを利用してプロシジャを作成できるが詳細については省略する．次にシェルプロシジャの簡単な例を示す．#で始まる行はコメント行である．

cshprocファイルの内容

```
# comment
date
who
ps
```

このcshprocプロシジャファイルはcshコマンドを用いて実行できる．

```
% csh cshproc
```

このシェルプロシジャの実行結果はdate, who, psの各コマンドの連続実行の結果と同じである．このシェルプロシジャファイルを実行可能なモード(2.4参照)に変更することにより一般のコマンドと同じように使うことができる．

```
% chmod a+x cshproc    ……実行可能ファイルにモード変更
% cshproc              ……コマンドのように実行できる
```

cshには3種類の自動実行シェルプロシジャファイルがある．これは通常，利用者のホームディレクトリに作成される．

(1) .login
(2) .cshrc
(3) .logout

.loginはログインする毎に最初に1回だけ実行される．このため利用者の開発環境に合わせて，ログイン時に1度だけ操作したいコマンドや特別な処理をしたいときに用いる．例えば，端末の設定をsttyコマンドで行ったり，サーチパス(シェルがコマンドを捜すディレクトリ)を追加したりすることができる．.cshrcはログイン時やシェルプロシジャの実行によってcshが呼び出される度に実行される．このファイルではローカルなシェル変数を設定することができる．また，.logoutは正常にログアウトされるときに実行されるファイルである．

3.5 ヒストリ機能

ヒストリ(history)機能とは過去に入力したコマンドをスタックしておいて，それを取り出して，そのまま，あるいは一部修正の上実行できることである．過去の何個まで記憶しておくかはhistory変数と呼ばれるシェル変数の設定で決まる．最近の5個を保存したければ,次のコマンドを直接入力するか，.cshrcファイル(.loginファイルでもよい)の中に置くことが必要である．

```
% set history=5
%
```

次のように，!マークの後ろに参照番号を入力することによりコマンドの再実行ができる．直前に実行したコマンドは!!で参照される．

```
%history
 23 cat letter
 24 mail a71234a <letter
 25 who
 26 ls -l
 27 history
% !25
a70001a
a70002a
:
```

数字の代わりに最も最近に実行されたコマンド名を入力してもよい．実際にはリスト中の先頭からの文字列をパターンマッチさせているだけであるから,曖昧でないなら先頭の1文字でもよい．すなわち，!25の代わりに，!wでもよい．また，以前投入したコマンドを一部修正の上実行できる．例えば，次のようにタイプミスしたとする．

```
% cal test.c
% cal: not found
```

直前のコマンドの修正実行は

```
% !!:s/元の文字列/修正文字列
```

または

```
% ^元の文字列^修正文字列
```

の形式で投入する．すなわち，今の場合は

```
% !!:s/l/t
```

とする．直前でない投入コマンドの場合は:(コロン)の後ろに部分置換の記述をする．例えば，

```
% !24:s/letter/letter2
```

は次のコマンドを投入したのと同じである．

```
% mail a71234a <letter2
```

:の後の部分置換の記述方法は上で述べた以外にもあるが，ここでは省略する．

3.6　コマンドの別名付け

単純なコマンド, 複合コマンドの区別なくコマンドに別名をつけることができる. こうして, よく用いられるコマンドが長いオプションや特定の文字列を引数として持っている場合に, それを省略して1つの名前で代表させることができる. 次の1つ目の例はls -lをlsの入力だけで実行するものである. 2つ目の例は現在のログイン名をwcへ引渡し, ログインユーザ数をカウントに出力する作業をwhonという文字列の入力で実行するものである.

```
% alias ls  ls -l
% alias whon who \|wc -l
% ls
total 30
drwxr-xr-x   2 a70001a  usr1          80 Jul 31 15:40 doc
-rw-r--r--   1 a70001a  usr1         287 Jul 10 19:34 test.c
-rw-r--r--   1 a70001a  usr1        2984 Jul 10 18:18 test.f
 :
```

\は|を単なる文字とみなす場合に用いる. 引数なしでaliasコマンドを実行するとすべての別名定義が表示される. また, unaliasコマンドを用いて別名定義を削除できる.

3.7 shとの比較

cshではプロンプトが%であったのに対してshでは$である. cshからshへの切り替えはshコマンドを入力すればよい. また, shからcshに戻るにはexitコマンドあるいは^Dを入力する.

```
% sh
$ exit
%
```

shとcshとでは使用できるコマンドに違いがあるほか, 次のような点が異なっている.

- シェル変数(環境変数)の記述が異なる.
  cshではhome, path, cdpath, mail, termのように小文字であるがshでは対応する変数はHOME, PATHなどのように大文字である.
- シェル変数の値の設定方法が異なる.
  cshでは次のようにset, あるいはsetenv, @が用いられる.
  ```
  % set prompt='!%!'
  ```
  shではset等がいらない.
  ```
  $ PS1='!$!'
  ```
- shの自動実行ファイルは.profileだけである. これはcshの.loginに相当する.
- シェルプロシジャのなかで実行の流れを変更する制御構文がcshではC言語風である.
- shにはヒストリ機能がない.
- shにはコマンドの別名付けの機能がない.

4. 利用者間の通信用コマンド

最後に利用者間の通信に便利なコマンドを紹介しておく．1つのUNIXシステム内の利用者間の通信に使うコマンドとしては，mail, write, mesgなどがある．

mailコマンドは電子メール機能を提供する．利用者が送受信するメッセージはシステム内のメールボックスと呼ばれるものを通して処理される．このメールボックスの機能によりログインしていない利用者にメッセージを送信することができる．これに対して，writeコマンドはログイン中の利用者に直接メッセージを送信する．このコマンドは送信相手の状況を無視して強制的にメッセージを送る．このため都合が悪いときにはwriteコマンドによる送信メッセージの受信拒否宣言をmesgコマンドで行えるようになっている．

mailコマンドは，引数に送信相手の利用者名(課題番号)を指定して，入力する．mailコマンド入力後メッセージを入力する．^Dが押されるまでが，送信メッセージとなる．

```
% mail a70007a a70008a
```

送信メッセージ

```
^D
%
```

mailコマンドは，標準入力から送信メッセージを入力するので，次のように，あらかじめletterファイルにメッセージを作成しておいて，そのファイルの内容をリダイレクションを用いて送信することもできる．

```
% mail a70007a a70008a <letter
```

一方，メール受信は，次のように行う．メールが到着していれば，ログインしたときに，

```
You have mail.
```

と表示される．到着したメールの内容を見るには，引数なしで，mailコマンドを入力すればよい．

writeコマンドは，ログインしている利用者に対して直接メッセージを送信する．送信したい相手がログインしているかどうかはwhoコマンドで調べればよい．writeコマンドによる送信もmailコマンドと同様にすればよい．

```
% write a70005a
```

送信メッセージ

```
^D
%
```

writeコマンドによるメッセージ送信は1行単位に行われるので，送信相手と会話的にメッセージのやり取りが行える．

writeコマンドによる自分の端末への割り込みを禁止するにはmesgコマンドを用いて，次のように入力する．

```
% mesg n
```

writeコマンドによるメッセージ受信を許可するには

```
% mesg y
```

とすればよい．

参考文献

1. 計算機マニュアル，UTS文法書(一般利用者コマンド編) V10L30 (24SP-1021-1)，富士通(株)．
   (英語版)：UTS User Reference Manual V10L30 (24SP-1021E-1)富士通(株)．
2. 計算機マニュアル，UTS文法書(プログラミング編) V10L30 (24SP-1031-1)，富士通(株)．
   (英語版)：UTS Programmer Reference Manual V10L30 (24SP-1031E-1)，富士通(株)．
3. 計算機マニュアル，UTS使用手引書(一般利用者編) V10L30 (24SP-1051-1)，富士通(株)．
   (英語版)：UTS User Guide V10L30 (24SP-1051E-1)，富士通(株)．
4. 計算機マニュアル，(英語版)：UTS Shell commands and Programming V10L30 (24GR-1110E-1)，富士通(株)．
5. 二村祥一：UTSの言語プロッセサとプログラム，九州大学大型計算機センター広報，Vol 20，No5，1987．