コマンドとアクセス権

目 次

1 シェル

1.1 シェル

シェル (shell) は，ユーザーがコマンド行に打ち込んだコマンド等を解釈し，コンピュータに実行させる役割をもつプログラムです[1] 。ユーザーがコンピュータにログインすると，そのユーザーのためにシェルが動き出します。これをログインシェル (login shell) といいます。ユーザーがコマンド行にコマンドをタイプして実行できるのは，シェルが動いているからです。

シェルには幾つかの種類があり，ユーザーがログインシェルとして何というシェルを用いるかは，予め管理者が設定しています。シェルが持つ機能や使い方はシェル毎に違いますので，`echo $SHELL` を実行して，利用中のシェルを確認しましょう[2]。/bin/tcsh と出力されますね。皆さんが使っているシェルは tcsh (ティーシーシェル) です。

1.2 コマンド入力の支援機能

tcsh が持つ機能の一例として，コマンド入力を助ける機能を幾つか紹介します。なお，ここで紹介する機能は，他の最近よく使われるシェルでも利用できます。

1.2.1 ヒストリの一覧と利用

過去に実行したコマンド行の履歴をヒストリ (history) といいます。Windows のコマンドプロンプトと同様に，tcsh でも上向きや下向きの矢印キーを使えば，過去に実行したコマンド行を再度使うことができます。

次のコマンドはヒストリ一覧を表示します。

```
history
```

ヒストリ一覧から，過去に実行したコマンド行を選んで再実行するには

`!`n

に続いて `<ENTER>` を押します[3]。ここで n はヒストリ一覧中のヒストリ番号です。

1.2.2 コマンド行の補完

コマンド名やファイル名の一部をタイプしてから `<TAB>` を押すことで，残りを自動的に補う (補完) ことができます。この機能を使うとコマンドやファイル名のタイプミスを防ぐこともできます。

1. [コマンド名の補完] コマンド行に `his` とタイプして `<TAB>` を押してみましょう。続いて `<ENTER>` を押しましょう。

[1] これは，シェル利用の一形態です。他の形態 (シェルスクリプトのためのコマンドインタプリタ) でシェルを利用することもあります。

[2] echo は引数をそのまま出力するコマンドですが，コマンド行における $ はシェルにとって特別な意味があるため，画面に $SHELL とは表示されません。

[3] 知っていて便利なヒストリ機能には次のものもあります。`!-`n (n だけ前に実行したコマンド行), `!!` (一つ前に実行したコマンド行 (`!-1` と同じ)), `!`*str* (*str* で始まる最も最近のコマンド行)。ここで n には正の整数を，*str* には適当な文字列 (通常はコマンド名またはその始めの一部) を記述します。これらに続いて `<ENTER>` を押すと，当該コマンド行の内容が実行されます。

2. [補完候補の表示] コマンド行に `hi` と打ってから `<TAB>` を押してみましょう。

   hi で始まるコマンドは複数ありますので，まだ補完はされません。

3. 続いて s をタイプしてコマンド行を `his` としてから `<TAB>` を押してみましょう。続いて `<ENTER>` を押しましょう。

4. [ファイル名の補完] まず，カレントディレクトリに存在するファイルの名前を ls コマンドを実行して確認してください。続いて，`cat` の引数に，既存のファイル名の一部をタイプしてから `<TAB>` を押してみましょう。ファイル名が補完されたら `<ENTER>` を押しましょう。

2 コマンドと検索パス

2.1 コマンドの実体

1. ディレクトリ /bin にどんなファイルがあるか調べましょう。`ls /bin` を実行してください。

   馴染みのある名前があるはずです。cat, cp, ls, pwd などのコマンド名が，ディレクトリ /bin の中のファイルの名前として表示されます。

2. `pwd` を実行して，pwd コマンドがどんなコマンドだったか，思い出しましょう。

3. 続いてコマンド行に `/bin/pwd` とタイプして`<ENTER>` を押してみてください。

   ファイル pwd の絶対パス名である/bin/pwd でも pwd コマンドを実行できましたね。

このことから予想されるとおり，pwd コマンドの実体は，ディレクトリ /bin に存在する通常のファイル pwd です。UNIX コマンドの多くは，コマンドと同名のファイルとして存在していて，/bin に存在するファイルはすべて実行可能なコマンドです[4][5]。

ファイルを扱うコマンドの引数にパス名が使えるのと同様に，コマンド自身を指定する際にもパス名が使えます。`/bin/pwd` で pwd コマンドを実行できたのは，そのためです。相対パス名でも可能です。

より正確にいうと，ファイルとして存在するコマンドを実行するには，本来，コマンドのパス名を使う必要があります。しかし，それでは不便ですので，/bin のような特定のディレクトリに存在するコマンドを，コマンド名のみで実行できるようにする仕組みがあります。そのおかげで，/bin/pwd や /bin/ls 等を `pwd` や `ls` とだけタイプして実行できるのです。

2.2 コマンド検索パス

2.2.1 コマンドの在処

実行しようとするコマンドがどこにあるのかを調べるために，which というコマンドが利用できます。which の引数にコマンド名を与えると，そのコマンドの所在が絶対パス名で表示されます。pwd, cal の各コマンドがどこにあるのかを調べるために

[4] /bin に存在するファイルはコマンドとして実行可能なバイナリ (binary) ファイルです。テキストファイルではないので，cat 等で中を読むことはできません。

[5] このことは Windows 等の他の OS 上で動作するコマンドやアプリケーションでも同様です。例えば，Web ブラウザ Firefox の実体は，Windows では通常 C:¥Program Files¥Mozilla Firefox¥firefox.exe です。

```
which pwd cal
```

を実行してください。

2.2.2 コマンド検索パス

コマンド行にコマンド名をタイプすると，シェルは実行すべきコマンド（ファイル）を特定のディレクトリから検索します。そのディレクトリは

```
echo $PATH
```

を実行して調べることができますので，試してください[6]。ディレクトリの絶対パス名が：(コロン) で区切られて表示されますが，先ほど which を使って調べたコマンドの在処は，すべてこのディレクトリのリスト中に含まれているはずです。

第 2.1 節で紹介したとおり，本来，pwd のようにファイルとして存在しているコマンドを実行するときには，パス名 (例えば pwd の絶対パスであれば /bin/pwd) を使う必要があります。しかし，それでは不便ですので，$PATH リスト[7]に含まれるディレクトリ内のコマンドについては，ファイル名 (コマンド名) のみで実行できるようになっているのです。逆に言えば，このリストに含まれるディレクトリ以外の場所にファイルとして存在するコマンドを実行するには，パス名を用いなければいけません。

ここで $PATH のリストに．(ドット = カレントディレクトリ) が含まれていないことに注意しましょう。コマンド名でコマンドを実行しようとすると，その実体であるファイルはあくまで $PATH リスト内のディレクトリから検索されます。したがって，**$PATH リストに．が含まれていなければ，カレントディレクトリ内に存在するコマンドを，コマンド名のみで実行することはできません**。そのため，仮にカレントディレクトリ内にコマンド *com* が存在するとして，それを実行したければ，カレントディレクトリに存在するコマンドであることを明示して `./com` 等とする必要があります[8] 。

なお，コマンド名でコマンドを実行する場合，$PATH リストの始めの方から，コマンドが探索されます。したがって，同名のコマンドが $PATH リストの中の複数のディレクトリに存在する場合，始めの方のディレクトリに存在するものが実行されます。

2.3 シェル組み込みコマンド

コマンド which はどこにあるのでしょう — `which which` を実行してください。シェルに入っているコマンド (shell built-in command) と表示されますね。このように，コマンドと同名のファイルではなく，シェルが内部に持っているコマンド（シェル組み込みコマンド）もあります。例えば，ログアウトするために使っている exit というコマンドは，正確には現在使っているシェルを終了するためのシェル組み込みコマンドです。exit でログアウトできるのは，ログインシェルを終了するとログアウトするからです[9]。

---

[6] echo は引数に与えた文字列を出力するコマンドですが，コマンド行に $ で始まる文字列を与えると，シェルがそれを別の文字列に置き換えてコマンドに渡しますので，$PATH とは表示されません。

[7] このディレクトリリストを単にパス (path) と呼ぶことが多い。

[8] コマンドの引数にファイル名を与えるときとの違いに注意してください。コマンド引数の場合，ファイル名のみを与えればカレントディレクトリ内のファイルを指定したことになりますので，例えば `cat` *file* を実行すれば，カレントディレクトリに存在するファイル *file* が cat コマンドで処理されます。

[9] こう言い切ることができない場合もあるのですが，細かい点は無視しましょう。

シェル組み込みコマンドは，コマンド検索パスの設定にかかわらず，コマンド名で実行できます。また，実行の優先順位は，コマンド名と同名のファイルとして存在するコマンドより高いです[10]。

シェル組み込みコマンドのマニュアルは，シェルのマニュアルに記載されています[11]。

2.4 参考 — コマンド検索パスへのディレクトリ追加

シェルとして tcsh を使っている場合

```
setenv PATH ${PATH}:dir
```

を一度実行すると，コマンド検索パスに *dir* を追加できます。ここで *dir* は，検索パスに追加したいディレクトリのパス名（普通は絶対パス名）です。

ただし，この設定はログアウトすると無効になりますので，ログインする度に行わなければなりません。それをせずに済ますには，ホームディレクトリのファイル.cshrc の最後に上記の内容を書き込んでおきます。.cshrc に書いた内容は，ログインする度に自動的に実行されます[12]。

なお，PATH はシェルの環境変数と呼ばれるものの一つであり，$PATH により環境変数 PATH に設定されている値を参照できます。また，setenv は環境変数に値を設定する tcsh の組み込みコマンドであり，setenv の一般的な書式は

```
setenv 環境変数名 値
```

です[13]。

2.5 問題

1. `/bin/ls` を実行してみましょう。
2. ディレクトリ /bin に存在するファイル ls を，ホームディレクトリの子ディレクトリ unix に複写しましょう。ただし，ファイル名を myls としてください。
3. カレントディレクトリをホームディレクトリとした状態で，unix に複写した myls を実行してみましょう。ディレクトリ unix はコマンド検索パスに含まれていないはずですから，ファイル myls のパス名を使って myls コマンドを実行する必要があります。
4. カレントディレクトリを unix に変更してから，カレントディレクトリを意味する .(ドット) で始まる相対パス名を使って myls を実行してください。
5. myls を削除してください。

---

[10] シェル組み込みコマンドの which に加え，ファイルの which コマンドも存在する場合があります。それを実行するにはパス名で /usr/bin/which 等と打つか，\which と打つ必要があります。シェル組み込みの which の方が実行の優先順位が高いからです。また man which で表示されるのは，シェル組み込みの which ではなく/usr/bin/which のマニュアルです

[11] `man tcsh`

[12] より正確には，csh や tcsh が起動する度に。

[13] Windows XP での PATH 設定法

1. スタート → 「マイコンピュータ」右クリック → 「プロパティ」で「システムのプロパティ」ウィンドウを開く。
2. 「詳細設定」タブをクリックして,「環境変数」ボタン を押す。
3. 「ユーザー環境変数」欄の中に PATH や path があれば，それをクリック してから「編集」ボタンを押す。さもなくば「新規」ボタンを押して「変数名」に PATH と記入する。

3　ファイルの詳細情報とアクセス権

3.1　ユーザーとグループ

UNIX システムの利用者は，そのシステム内で重複しないユーザー名 (ログイン名) でシステムに登録されている必要があります。利用者は正しいユーザー名とパスワードを入力することで，システムの利用が許可されます (ログイン)。

UNIX にはユーザーに加えて **グループ** *(group)* という概念があります。通常，UNIX のシステムには複数のグループが登録されており，各ユーザーは少なくともそのうちの一つに属しています。自分が所属するグループは

```
groups
```

というコマンドで調べることができます。試してみましょう。

3.2　ファイルの詳細情報 — ls -l

UNIX のファイルには，ファイル名の他にも種々の情報が付与されています。ls にオプション `-l` (long format) を付けて実行することにより，ファイルの詳細情報を表示することができます。

また，ls の引数にディレクトリを指定できることは紹介済みですが，ls の引数には通常のファイルの名前を与えることもできます。その場合，ls は与えられたファイルの情報のみを出力します。複数のファイル名を与えることも可能です。

さて，カレントディレクトリに通常のファイル year が存在するとします。このとき

```
ls -l year
```

を実行すると，ファイル year の詳細情報を得ることができます[14]。次に示すのは，ユーザー skonno による実行例です。本節では，アクセス権とリンク数を除く各項目について説明します[15]。

```
種類          リンク数 グループ          更新日時
|           |        |              <---------->
-rw-r--r--  1 skonno student 2007  7月 13 09:21 year
 <------->    |              |                    |
 アクセス権   所有者          サイズ               ファイル名
```

種類 ファイルの詳細情報における一番左の記号 (-) はファイルの種類を表します。記号の示す意味は次のとおりです[16]。

- -: 通常のファイル
- d: ディレクトリ

所有者 ファイルを所有するユーザー (所有者; owner) の名前です。デフォルトでは[17]，ファイルを作成したユーザーが，そのファイルの所有者です。

---

[14] 引数にファイル名やディレクトリ名を与えなければ，カレントディレクトリ内のファイルに関する詳細情報一覧が得られます。

[15] リンク数については，この授業では扱いません。

[16] ファイルの種類はこの他にもありますので，この欄が - や d 以外になることもあります。

[17] 「特に何もしなければ (by default)」

グループ この項目はファイルの所属グループを表します[18] 。ファイルに所有者があるのと同様に，ファイルは第 3.1 節で説明したグループにも属します。デフォルトでは，ファイルの所有者が所属する第一番目のグループが，そのファイルの所属グループとなります。

サイズ byte 単位でのファイルサイズです。テキストファイルの場合，1 文字が 1 byte です。ただし，日本語の文字では，いわゆる半角文字を除き，1 文字あたり 2 byte です。

更新日時・ファイル名 文字通り，ファイルの最終更新日時とファイル名です。

3.3 ファイルのアクセス権

UNIX のシステムにはただ一つのディレクトリ／ファイルの階層構造が存在し，各ユーザーのホームディレクトリは，すべてその中にあります。ですから，他の人のホームディレクトリの中を覗いたりファイルの内容を見たり，さらにはファイルやディレクトリを削除することもできそうです...それでは困りますね。

このようなことを防ぐために，UNIX ではファイルやディレクトリに関する様々な権限を，ファイルの所有者や，ファイルが属するグループ内のユーザーなどに対して設定するための仕組みがあります。それらの権限をファイルの**アクセス権** (permission) といいます。アクセス権のことをパーミッションやアクセス許可，使用許可等と呼ぶこともあります。

3.3.1 アクセス権の読み方

ファイルのアクセス権は，`ls -l` で表示されるファイルの詳細情報から調べることができます。そのために必要な情報はアクセス権と所有者およびグループです。本節では，第 3.2 節で例示したファイル year の詳細情報

アクセス権: `rw-r--r--`，所有者: skonno，グループ: student

を例に，アクセス権の読み方を説明します。

1. アクセス権を構成する 9 文字を 3 文字ずつの組に分けます。year の場合，`rw-` と `r--` と `r--` の 3 組です。

   各組は左から順に

   (a) ファイルの所有者 (skonno) に対するアクセス権 (`rw-`)

   (b) ファイルのグループ (student) に属する，所有者 (skonno) 以外のユーザーに対するアクセス権 (`r--`)

   (c) 上記以外のユーザーに対するアクセス権 (`r--`)

   を表しています。

2. 3 つの組それぞれについて，各組を構成する 3 つの文字からファイルのアクセス権を調べます。3 つの文字は左から順に，次に示す 3 種類のアクセス権を表しています。

[18] グループを表示するために，ls にオプション `-g` が必要な UNIX システムもあります

(a) ファイルの内容の読み取り権[19](r または -)

`r`：読み取り可能 (readable)

`-`：読み取り不可能

(b) ファイルへの書き込み権[20](w または -)

`w`：書き込み可能 (writable)

`-`：書き込み不可能

(c) ファイルの実行権[21] (x または -)

`x`：実行可能 (executable)

`-`：実行不可能

以上より，year のアクセス権は

1. 所有者 skonno に対して，読み取り可・書き込み可・実行不可 (`rw-`)
2. グループ studnet に属する，skonno 以外のユーザーに対して，読き取り可・書き込み不可・実行不可 (`r--`)
3. その他のユーザーに対して，読み取り可・書き込み不可・実行不可 (`r--`)

であることがわかります。すなわち，year の所有者 skonno は，このファイルの内容を読んだり，ファイルに書き込みを行うことができます。グループ studnet に属する skonno 以外のユーザーと，その他のユーザーは，このファイルの内容を読み取ることのみが許されています[22]。

3.3.2 アクセス権の変更 — chmod コマンド

ファイルの所有者は，chmod (change file mode) コマンドを使って，ファイルのアクセス権を変更することができます。chmod の基本的な使い方は

`chmod` *mode file ...*

であり，*mode* には，*file ...* の現在のアクセス権をどのように変更するのかを指定します。*mode* に指定する要素は「誰に」「どの権限を」「与える (または取り除く)」の三つですが，記述の順番は次のとおりです。

1. 「誰に」：u, g, o, a の何れかを指定します[23]。
   - u：ファイルの所有者 (user[24])
   - g：所有者を除くファイルのグループに属するユーザー (group)
   - o：その他のユーザー (other)
   - a：u, g, o の全て (all)

[19]ディレクトリの場合は，ディレクトリ内のファイル一覧を得る権限
[20]ディレクトリの場合は，ディレクトリ内にファイルを作ったり，ファイルを削除する権限
[21]ディレクトリの場合は，ディレクトリ内のファイルにアクセスする権限 (検索権)
[22]ただし，実際にファイルの内容を読めるかどうかは，そのファイルが存在するディレクトリのアクセス権にも依存します
[23]u, g, o を組み合わせるなど，より複雑な記述も可能ですが，何れか一つを指定する方法を知っていれば困らないでしょう。詳細を知りたければ man chmod 等で調べてください。
[24]owner と呼ぶべきですが，other と区別するために user と記します

2. 「与える」または「取り除く」: + か - を指定します。
   - +: 与える
   - -: 取り除く
3. 「どの権限を」; r, w, x の何れかを指定します。
   - r :読み取り権
   - w :書き込み権
   - x :実行権

例えば

`chmod g+w` *file*

だと，*file* のグループに所属する (所有者以外の) ユーザーに，書き込みの権限を与えることになります。また，

`chmod a-w` *file*

では，全てのユーザーに対して，file への書き込みを禁止します。

3.3.3 発展 — アクセス権の 8 進数指定（絶対指定）

chmod における *mode* を `g+w` などの記号で指定する代りに，数 (8 進数) で指定することも可能です。数による指定では，まず次の表にあるように，読み取り・書き込み・実行の可または不可を表す 3 文字の記号を，- のときは 0 に，それ以外のときは 1 に対応させます。そうしてできあがった 0 と 1 の並びを 3 桁の 2 進数として解釈し，8 進数で表現します。

記号	0,1 の並び (2 進数表記)	8 進数	意味
`---`	000	0	読取不可・書込不可・実行不可
`--x`	001	1	読取不可・書込不可・実行可
`-w-`	010	2	読取不可・書込可・実行不可
`-wx`	011	3	読取不可・書込可・実行可
`r--`	100	4	読取可・書込不可・実行不可
`r-x`	101	5	読取可・書込不可・実行可
`rw-`	110	6	読取可・書込可・実行不可
`rwx`	111	7	読取可・書込可・実行可

上記の一桁の 8 進数を，所有者・グループ・その他のユーザーに対するアクセス権として順に並べると，アクセス権の 8 進数表記ができあがります。この方法による chmod の例を次に示します.

- `chmod 644` *file* (すべてのユーザーが読み取り可能，所有者のみ書き込み可能)
- `chmod 755` *file* (すべてのユーザーが読み取りと実行可能，所有者のみ書き込み可能)

この方法では，現在のアクセス権とは無関係に，指定したアクセス権が設定されることになります (絶対指定)。

3.4 問題

1. ホームディレクトリに存在するファイルの詳細情報を一覧表示して，観察してみましょう。通常のファイルとディレクトリの違いや更新日時，所有者，アクセス権等を確認してください。

2. ファイル /bin/pwd や /bin/ls の詳細情報を表示してください。適切なアクセス権になっていることが確認できますか？

3. `ls /bin` と `ls -F /bin` の出力を比較してください。実行可能なファイルを表す印は何でしょう?。

4. ホームディレクトリに存在するディレクトリ unix に，カレントディレクトリを変更してください。

   次に，空行（改行）が一つだけ入ったファイル now を作ります。ここでは echo コマンドとリダイレクトを使って

   ```
   echo > now
   ```

   を実行しましょう。既に now が存在したら，中身は上書きされて改行だけになります。

   ファイル now の詳細情報を表示して，アクセス権やファイルサイズ，更新時刻などを確認してください。

5. 現在の日付や時刻をファイル now に書き込みましょう。date コマンドとリダイレクト (`>` または `>>`) を使ってください。

6. now のファイルサイズや更新時刻などを確認しましょう。now の内容も見てみましょう。

7. chmod コマンドを使って now のアクセス権を `-w-r--r--` に変更してください。(ファイル所有者の読み取り権を外します)

8. now の中を cat コマンドで見てみましょう。

   見えないはずですね。

9. `date >> now` で now に書き込みをしてみましょう。

   これは可能ですね。

10. now のアクセス権を `r--r--r--` に変更してから (ファイル所有者に読み取り権を与え，さらに書き込み権を外します)，中身を見てみましょう。

11. 再度，now に現在の日付や時刻を書き込みましょう。

    できませんね。書き込み権を外すと，誤操作からファイルを保護できます。

12. now の削除を試みましょう。

    書き込み許可のないファイルは，すぐには削除されず，

    rm: remove write-protected 通常ファイル ‘now’?

    のような表示がでます。「書き込み保護されているファイルを削除しますか？」という確認メッセージです。ここでは `n` で答えてください。ファイルは削除されません。

13. now のアクセス権を `rw-------` に変更しましょう。

    ファイルの所有者以外の人は now を読み書きできなくなりました。一方，所有者は now の内容を見ることも変更することもできます。

14. now を削除しましょう。

    今度はうまくいきましたね。

15. 自分の生まれた月のカレンダーを表示するコマンドを作って，実行します。

    複雑な処理をするコマンドを作成するにはプログラミングの知識が必要ですが，簡単な処理のコマンドは，既存のコマンドをテキストファイルに記入し，そのファイルに実行の許可を与えるだけで，作成することができます。

    以下の手順に従って作業をしてください。

    (a) 次のコマンドを順に実行して，結果を確認してください。

        `echo` *name* `was born in`
        `cal` *month year*

        ここで，*name* には自分の名前を，*month year* には生まれた月と年を与えてください。

    (b) 前項でコマンド行にタイプした文字列 (echo ... と cal ...) 2 行が入ったテキストファイル mybirth を作成してください。echo コマンドと，`>` や `>>` を使いましょう。コマンドの実行履歴（ヒストリ）も活用するといいでしょう。

    (c) そのファイルに，すべてのユーザーが実行可能な実行権を与えてください。

    (d) ファイルに実行権を与えましたので，ファイルは mybirth という名のコマンドになりました。

        mybirth はカレントディレクトリにありますが，カレントディレクトリはコマンド検索パスに含まれていません。このことに注意して，作成したコマンド（ファイル）を実行してください。