５ＭＳ実験ＯＳ　第２回　シェル変数，プロセス

１．gedit（テキストエディタ）

日本語（SHIFT_JIS コード）の文字化け対策。

以下のコマンドを実行する。

```
gsettings set org.gnome.gedit.preferences.encodings auto-detected "['UTF-8'(改行なし)
,'CURRENT','SHIFT_JIS','EUC-JP','ISO-2022-JP','UTF-16']"
```

gedit の起動コマンド

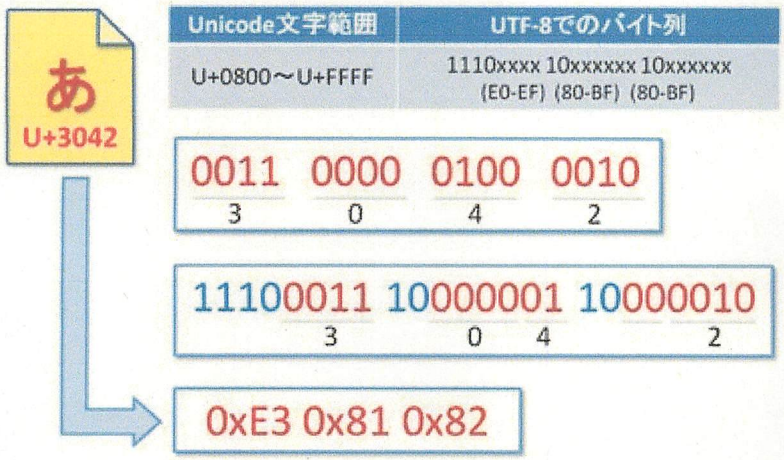
`gedit[E]`とコマンド入力すると起動する。`gedit ファイル名[E]`でファイルを指定して起動する。

２．文字コード

世界中の様々な文字の集合体が「文字集合」と言い、文字集合で定義した個々の文字をコンピューター上で表示する数値の振り方が「文字符号化方式」と言います。

ファイル保存や HTTP で指定する文字コードは「文字符号化方式」（Shift_JIS　EUC-JP(Unix 系 OS)　ISO-2022-JP(電子メール)　UTF-8　UTF-16 など）UTF-8 は、ASCII コードとの互換性が高いことから、世界中の多くのソフトウェアが用いています。幅広く普及していることを考えると、UTF-8 は世界的にもポピュラーな文字コードだと言えるでしょう。

図1 「あ」のUTF-8でのエンコード方法




文字コード確認，変換

nkf のインストール（sudo はスーパーユーザ root 権限で，コマンドを実行する。）

```
sudo apt-get update
sudo apt-get install nkf
```

レポート 2-1：`gedit` で適当な日本語の文章を入力した `moji.txt` ファイルを作成し，`nkf --guess ファイル名[E]`で文字コードを確認せよ。

EUC-JP へ変換

```
$ nkf -e --overwrite <textdata>
```

Shift-JIS へ変換

```
$ nkf -s --overwrite <textdata>
```

UTF-8 へ変換

```
$ nkf -w --overwrite <textdata>
```

## 3. 環境変数，シェル変数

標準的な UNIX の変数は、環境変数とシェル変数、ふたつのカテゴリに分類されます。 大まかな説明として、シェル変数は一時的な作業の為に現在のシェルで使用されるものです。 環境変数はより重要な意味を持ち、ログイン時にセットされてログイン中は常に有効となります。慣例的に、環境変数の名前には大文字が用いられ、シェル変数には小文字が用いられます。

シェル変数は現在実行中のシェルだけで有効な変数ですが，環境変数はシェルから実行したコマンドにも引き継がれる変数です。

環境変数の例

SHELL（使用するシェル）

USER (あなたのログイン名)

HOME (あなたのホームディレクトリのパス)

HOST (使用中のコンピュータの名前)

ARCH (コンピュータのプロセッサのアーキテクチャ)

DISPLAY (X ウィンドウを表示するコンピューターの画面の名前)

PRINTER (プリントジョブを送信するデフォルトのプリンター)

PATH (シェルがコマンドを検索するべきディレクトリ)

LANG（ユーザーの使用言語を設定する。この値を見て処理を変えるプロセスがある。）

レポート 3-1 : `echo $SHELL[E] echo $HOME[E] echo $PATH[E] echo $LANG[E]`それぞれのコマンドの返答を書け。

環境変数の追加，変更

HOGE という環境変数を追加する。

レポート 3-2 : `echo $HOGE[E]` 返答を書け。

`export HOGE=aiueo[E] echo $HOGE[E]` 最後のコマンドの返答を書け。

環境変数をログインする度に手で入力することはありません。~/.bashrc や ~/.bash_profile といったファイルで export 文を上のように書いておけば良いのです。

レポート 3-3 : `.bashrc` ファイルで `alias` 命令によって `la` がどのようなコマンドが実行されるように設定されているか。

## 4. シェルスクリプト

レポート 4-1 : 以下を記述した実行権限のあるファイル `sh1.sh` を作成し，`./sh1.sh[E]`により実行する。実行結果を書け。

```
#!/bin/bash
echo "Hello, World!"
```

### ＊変数，引数

レポート 4-2 : 以下を記述したファイル実行権限のある `sh2.sh` を作成し，`./sh2.sh aaa[E]` により実行する。実行結果を書け。

```
#!/bin/bash
var1="変数"
echo "$var1 $1"
```

レポート 4-3：ディレクトリを作り，そこにファイル名 apple1 apple2 apple3 の 3 個のファイルを作る。その後以下のシェルスクリプトを作成し，実行する。ファイル名はどうなるか。

```
#!/bin/bash
for f in apple*
do
mv $f ${f/apple/orange}
done
```

## 5． プロセス

プロセスとは、「実行中のプログラム」だと思えばいい。また、一つのプログラムから複数のプロセス(プログラム)が生成されることも多くある。

OS は、プログラムに適切なコンピュータ資源、メモリや CPU 時間などを割り振ることになる。割り振られたリソースの上でプログラムはオペレーティングシステム上のプロセスとして稼動することになる。プロセスは基本的に役目を果すと終了して、メモリなどの資源を開放する。

ps コマンド：実行中のプロセス情報を表示する。

レポート 5-1：ps[E] の返答を書け。

レポート 5-2：ps aux[E]ですべての起動中プロセスが表示される。現在起動中のプロセスの個数を書け。

pstree コマンド：実行中のプロセス親子関係を表示する。

レポート 5-3：gnome-terminal の子プロセスを書け。

レポート 5-4：gedit[E] 新しい端末を起動し，そこで kill gedit のプロセス ID[E] でどのようなことが生じるか。

/proc/<プロセス ID>にある情報

レポート 5-5：gedit&[E] cd /proc/gedit のプロセス ID[E] ls[E] cat status[E] 最後の返答で表示される Uid:の行を書け。

## 6． ジョブ

* Linux の「ジョブ」とは？

コンピュータに実行させる“作業のまとまり”を「ジョブ」と呼びます。例えば、コマンドラインでは「コマンドを入力して、[Enter] キーを押す」と、1 つのジョブが実行されることになります。

ただし、1 つのジョブで実行されるのは、1 つのコマンドとは限りません。コマンドラインで「ls」を実行した場合は ls コマンドだけが実行され、これで 1 つのジョブになります。「ls | more」と実行した場合は、ls コマンドと more コマンド、2 つのコマンドが実行されますが、これも 1 つのジョブになります。

「バックグラウンドジョブ」とは？

コマンドプロンプトでコマンドを入力して実行すると、通常はそのコマンドの処理が終了するまで次のプロンプトが表示されません。

* バックグラウンド

そこで、コマンドを「バックグラウンド」で実行させるようにすれば、すぐに次のコマンドを実行できるようになります。コマンドをバックグラウンドで実行させるには、コマンドの最後に「&」を付けて実行します。「gedit&」と入力して起動した場合を見てみましょう。gedit が起動しますが、プロンプトも表示されており、コマンドが入力可能な状態になっています（画面 3）。画面内に表示されている「[1]」が「ジョブ番号」で、続く「5496」が「プロセス ID」。なお、ジョブ番号は、シェル（今回の場合は bash）が「自分自身が実行しているジョブ」に付けている番号のことです。

＊　;と&&

「firefox & gedit &」の指定では、Firefoxの起動に続いて、すぐにgeditが起動しました。「&」を「;」に変えて「firefox ; gedit ;」とすると、“Firefoxが終了してからgeditが起動する”という動作となります。コマンドラインで「./configure && make」と実行した場合は、「configureを実行して、それが成功したらmakeを実行する」という内容になります。

レポート6-1：**(ls;sleep 30;ls)&[E]**を実行し，スリープ中に**jobs[E]**を実行したときの返答を書け。また**(ls;sleep 30;ls)&[E]**はどのような動作をする。

＊　シグナル

「シグナル」はプロセスとプロセスの間で通信を行う際に使用される“信号”のことで、シグナルを受け取ったプロセスは“何らかの動作”を行います。その動作は、例えば「再起動」であったり、「終了」であったりします。［CTRL］＋［C］は「SIGINT」というシグナルを行うキー操作、［CTRL］＋［Z］は「SIGTSTP」というシグナルを行うキー操作です。

| | | | |
|---|---|---|---|
| SIGHUP | 1 | Term | 制御端末の切断（ハングアップ）、仮想端末の終了 |
| SIGINT | 2 | Term | キーボードからの割り込みシグナル（通常は［CTRL］＋［C］） |
| SIGKILL | 9 | Term | 強制終了シグナル（KILLシグナル） |
| SIGALRM | 14 | Term | タイマー（Alarm）による終了 |
| SIGTERM | 15 | Term | 終了シグナル（「kill」コマンドのデフォルトシグナル） |
| SIGCONT | 18 | Cont | 一時停止しているジョブへの再開シグナル |
| SIGTSTP | 20 | Stop | 端末からの一時停止シグナル（通常は［CTRL］＋［Z］） |
| SIGTTIN | 21 | Stop | バックグラウンドジョブ／プロセスのキーボード入力待ち |

動作

| | |
|---|---|
| Term | プロセスの終了 |
| Ignore | 無視 |
| Core | プロセスの終了とコアダンプ出力 |
| Stop | プロセスの一時停止 |
| Cont | プロセスが停止中の場合は実行の再開 |

端末で使用するコマンドの多くは、SIGINTを受け取ると実行を終了するようになっているため、［CTRL］＋［C］キーで「実行中のプログラムを終了」させることができます。一方で、「less」コマンドのように終了しないようになっているコマンドもあります。

端末で実行しているコマンドは、SIGHUPを受け取ると終了します。従って、X（X Window System）環境でコマンドを実行している最中に端末アプリケーションを終了すると、そのコマンドは終了します。

レポート6-2：端末を2個開く。一方で**ping localhost[E]**，もう一方で，**ps -a -O stat[E]**でプロセスを確認した後，**killall -STOP ping[E] killall -CONT ping[E] killall -INT ping[E]** を順に実行する。各コマンド実行時どのようになるか。

*trap コマンド

trap コマンドは送出されたシグナルを捕捉し、あらかじめ指定されていた処理を実行するコマンドである。

trap ’コマンド’ シグナルリスト

レポート6-3：**trap 'echo trapped' 2[E]** その後［CTRL］＋［C］を押す。どのようになるか。
さらに，その後**trap 2[E]** その後［CTRL］＋［C］を押す。どのようになるか。

レポート6-4：**trap**の動作を確認する簡単なシェルスクリプトを作成し，動作確認せよ。

## 7. 調査

レポート 7-1：UNIX(Linux)のプロセス管理（プロセスの状態遷移についても）について書け。A4，1 ページ程度にまとめよ。