プログラミング応用

http://bit.ly/ouyou3d

オペレーティングシステムとプログラミング（2）

前期 第4週

2018/5/8

本日は・・・

引き続き、UNIXコマンドによるOS機能の利用

- **コマンドへの入出力**
- **複数コマンドの連携**
  - **パイプ**
  - **シェルスクリプト**

本日は・・・

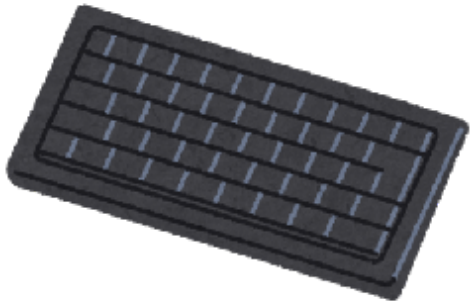
引き続き、UNIXコマンドによるOS機能の利用

- コマンドへの入出力
- 複数コマンドの連携
  - パイプ
  - シェルスクリプト

標準入出力

標準入力から入力

標準入力



標準入力から
コマンドを入力

プログラム

```
scanf(“%d”, &x);
printf(“%d”, x);
```

UNIXコマンド

```
$ ls
week01.c   week02.c
```

標準出力へ出力

標準入力

標準出力へ
結果を出力

多くのコマンドは、入力元/出力先を指定しない場合

標準出力/標準入力が用いられる

リダイレクトとは

入力元/出力先をファイルに変更する機能

- ファイルに保存

```
$ ls > output.txt
```

（lsコマンドの出力結果を、output.txtというファイルに保存）

- ファイルに追記

```
$ ls >> output .txt
```

（lsコマンドの出力結果を、output.txtというファイルに追記）

リダイレクトで便利なコマンド

- echo ・・・ 文字列を標準出力するコマンド

```
$ echo “This is a test.”
```

（This is a test. という文字が端末に表示される）

- cat ・・・ ファイル内の文字列を標準出力するコマンド

```
$ cat test.txt
```

（ファイルtest.txtの内容が端末に表示される）

【例1】

```
$ echo “This is a test.” > test.txt
$ cat test.txt > test2.txt
```

本日は・・・

引き続き、UNIXコマンドによるOS機能の利用

- コマンドへの入出力
- 複数コマンドの連携
  - パイプ
  - シェルスクリプト

パイプとは

複数コマンドを組み合わせて使う機能

【使い方】
複数のコマンドをパイプ記号（|）で結合

```
$ コマンドA | コマンドB
```

（コマンドAの出力がコマンドBの入力となる）

パイプの使用例

【例2】

```
$ ls -l | grep “.c”
```

（lsコマンド出力がgrepコマンドの入力となる）

- lsコマンドに「-l」オプションを付けると、詳細情報が付いたリスト形式でファイル一覧が出力される
  - test.c
  - memo.txt
  - sample.py
- grepコマンドは特定文字を含む行み出力する
  - test.c
  - ~~memo.txt~~
  - ~~sample.py~~

本日は・・・

引き続き、UNIXコマンドによるOS機能の利用

- コマンドへの入出力
- 複数コマンドの連携
  - パイプ
  - シェルスクリプト

シェルスクリプトとは

複数のUNIXコマンドを組み合わせて
書くためのプログラミング言語

- スクリプト言語の1つ
  （機械語へのコンパイルが不要）
- C言語と同じように、変数, 数値演算, 選択構造（if文など）, 繰り返し構造（for文など）が利用可能

シェルスクリプトの作成

- ファイルを作成する（拡張子は「.sh」にする）

```
$ gedit test.sh &
```

- 以下ようなUNIXコマンドをファイルに記述する
  （上から記述したコマンドが逐次実行される）

【例3（test.shの記述内容）】

```
#!/bin/sh
mkdir week5
cd week5
echo “test” > test.txt
```

シェルスクリプトの実行

- ファイルに実行権限を与える

```
$ chmod +x test.sh
```

- プログラムを実行する

```
$ ./test.sh
```

シェルスクリプトの変数

- 変数定義 ・・・ C言語とは違って、型の指定は不要

  【例】文字列を代入

  ```
  var1="This is a test."
  ```

  数値を代入

  ```
  var2=10
  ```

- 変数参照 ・・・ 「${変数名}」で変数を参照する

  【例】Test.txtという空ファイルを作る

```
variable3="Test"
touch ${variable3}.txt
```

シェルスクリプトの数値演算

- 積以外の演算子はC言語と同様
  - a + b（aとbの和）
  - a - b（aとbの差）
  - a * b（aとbの積）
  - a / b（aとbの商）
  - a % b （aとbの剰余）
- 計算結果を変数に代入するにはexprを用いる
  - number1 = `expr 1 + 2`
  - （1+2の結果を変数number1に代入）

シェルスクリプトの数値演算

【例4】

```
#!/bin/sh
number1 = `expr 1 + 2`
echo ${number1}
```

- 変数number1に1+2の計算結果を代入（バッククォート「`」に注意）
- echoコマンドで変数number1の値を表示

代入演算子（=）の前後には空白を入れない

シェルスクリプトのif文

【if文】

```
if 条件式 ; then
  処理1
fi
```

シェルスクリプトのif文

【if～else文】

```
if 条件式 ; then
   処理1
else
   処理2
fi
```

シェルスクリプトのif文

【if～elif～else文】

```
if 条件式1 ; then
  処理1
elif 条件式2 ; then
  処理2
else
  処理3
fi
```

シェルスクリプトのif文

if [${number1} -eq 5]; then

- [の後と] の前には空白を入れる
- 数値の比較は「-eq」
  - -ne → 等しくなければ真
  - -gt → 大きければ真
  - -lt → 小さければ真
  - -ge → 以上であれば真
  - -le → 以下であれば真

【例5】

```
#!/bin/sh
number1=5
if ${number1} == 5; then
  echo “The number is 5.”
else
  echo “The number is not 5.”
fi
```

- number1に数字を代入
- number1に格納された数字が5なら「The number is 5.」と出力

シェルスクリプトのfor文

【for文】

```
for 変数 in 値リスト

do
   処理

done
```

シェルスクリプトのfor文

【例6】

```
#!/bin/sh
for i in 1 2 3 4 5 6 7 8 9 10
do
   echo ${i} > ${i}.txt
done
```

- まずiに1が代入されて処理され、次にi2が代入されて処理され･･･
  と繰り返される
- ファイル「1.txt」に「1」が書き込まれ、「2.txt」に「2」が書き込まれ･･･「10.txt」に「10」が書き込まれる

シェルスクリプトのfor文

【応用例】

```
#!/bin/sh
for file in `ls *.c`
do
   echo ${file}
done
```

- 値リストはコマンドを用いて取得することも可能
- 「拡張子が.c」のファイル名が値リストとなり、echoコマンドが繰り返し実行される

【練習4-1】

リダイレクトの例1と、パイプの例2をそれぞれ実行して、結果を確認してみましょう。

【練習4-2】

シェルスクリプトの例3～例6のプログラムを作成して、それぞれの実行結果を確認してみましょう。

【課題4-1】

講義資料ページから「cat.txt」をダウンロードし、~~echoコマンドとリダイレクトの機能を用いて、~~「吾輩猫である」の全文を標準出力に表示して下さい。

catコマンドを用いて

【課題4-2】

cat.txtのうち「猫」を含む行のみを表示するシェルスクリプトを「print_cat.sh」という名前で作成して下さい。「print_cat.sh」に実行権限を与えて、実行結果を確認して下さい。

【課題4-3】

パイプ機能を使って、「吾輩猫である」の文のうち「猫」を含む行数を数えるシェルスクリプト（count_cat.sh）を作成して下さい。

- 文字列行数をカウントするには、wcというコマンドありますので、使い方を調べてみましょう。
- echo（もしくはcat）, grep, wcコマンドをパイプでうまく組み合わせた記述のシェルスクリプトを作ってみましょう。

【課題4-4】

10+20の計算結果をechoコマンドで出力するシェルスクリプト（calc.sh）を作成して下さい。

【課題4-5】

1から10の間の偶数を標準出力するシェルスクリプト（even_number.sh）を作成して下さい。