プログラミング応用

第5週

前回の内容

- UNIXコマンドを用いたOS機能の利用
  - UNIXコマンドを用いたプロセス管理、空間管理
    - ps(起動中プロセスの表示)
    - pstree（プロセスを木構造で表示）
    - top（プロセス、メモリを監視）
    - free（メモリの使用状況を表示）
  - UNIXコマンドを用いたファイル管理
    - mkdir
    - cp
    - mv
    - rm

本日の講義

- UNIXコマンドによるOS機能の利用（続き）
  - 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

本日の講義

- UNIXコマンドによるOS機能の利用（続き）
  - 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

- lsコマンド等の出力は通常「端末」に表示
- 標準出力と標準入力:
  - 演習室のシステム(UNIX)では以下のように決まっている
    標準出力:端末の画面
    標準入力:キーボード

    多くのコマンドでは入力元/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(1/2)

- リダイレクト:
  入力元/出力先をファイルに変更する機能

  $ 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

  $ 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(2/2)

- echoコマンド
  - 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コマンド
  - ファイル内の文字列を標準出力するコマンド

- $ echo “Test” > test.txt
  $ cat sample.txt > test.txt

- demo

本日の講義

- UNIXコマンドによるOS機能の利用（続き）
  - 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

パイプの使い方

- パイプ：複数のコマンドを組み合わせて使う機能
  - 複数のコマンドをパイプ記号（|）で結合

- 例）
  - $ ls –l | grep “.c”
    lsコマンドの出力をgrepコマンドに送る

パイプの動作例

- $ ls –l | grep “.c”の動作
  - lsコマンドに-lオプションを付けるとファイル一覧をリスト形式で表示

    aaa.c
    bbb.txt
    ccc.txt

  - grepコマンドは特定の文字を含む行のみ出力するコマンド

    aaa.c
    ~~bbb.txt~~
    ~~ccc.txt~~

- demo

本日の講義

- UNIXコマンドによるOS機能の利用（続き）
  - 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

- 複数のUNIXコマンドを組み合わせて使うプログラミング言語
  - スクリプト言語の一つ（機械語へのコンパイル手順が不要）
  - C言語と同じように変数、数値演算、制御構文（if文など）、繰り返し（for文等）が利用可能

シェルスクリプトの作成

- ファイルの作成（拡張子を.shに設定）
  $ gedit test.sh &

- 以下のようにUNIXコマンドをファイルに記述
  – 上の方に記述したコマンドから逐次実行される

```
mkdir week5
cd week5
echo “test” > test.txt
```

シェルスクリプトの実行

- 実行権限を与えて実行
  - 実行権限を与える
    $ chmod +x test.sh
  - 実行
    $ ./test.sh

シェルスクリプトの変数

- 変数の定義
  - C言語と違って型の指定は不要
    例) 文字列を代入
    variable1=“This is a test.”
    数値を代入
    variable2=10
- 変数の参照
  - ${変数名}という書式で変数を参照する
    例）Test.txtという空のファイルを作る
    variable3=“Test”
    echo “Test” > ${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 - ba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という記述を用いる
  - number1=`expr 1 + 2`

シェルスクリプトの数値演算

- 記述例
  - 変数number1に1+2の計算結果を代入
  - echoコマンドで変数number1の値を表示

```
number1=`expr 1 + 2`
echo ${number1}
```

シェルスクリプトのif文

- 記述方法の基本(if)

```
if 条件式 ; then
  処理
fi
```

シェルスクリプトのif文

- 記述方法の基本(if-else)

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

シェルスクリプトのif文

- 記述方法の基本(if-elif-else)

```
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

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

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

シェルスクリプトのfor文

- 記述方法の基本

```
for 変数 in 値リスト
do
   処理
done
```

シェルスクリプトのfor文

- 記述例

```
for i in 1 2 3 4 5 6 7 8 9 10
do
  echo “${i}” > ${i}.txt
done
```

for文の応用例

- 値リストはコマンドを用いて取得することも可能

```
for file in ‘ls –l *.c’
do
  echo “${file}”
done
```

演習1(リダイレクト)

- 講義資料ページからcat.txtをダウンロードし、echoコマンドを用いて、「吾輩は猫である」の全文を標準出力に表示せよ

演習2（シェルスクリプトの作成）

- 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という名前で作成せよ
- print_cat.shに実行権限を与え実行せよ
- 実行を確認したらadd/commit/pushし、GitHubに提出せよ

演習3（パイプ）

- 「吾輩は猫である」の文のうち「猫」を含む行の数を数えるシェルスクリプト(count_cat.sh)を作成し提出せよ
  - ヒント1：
    文字列の行数をカウントするwcというコマンドがあるので使い方を調べよ
  - ヒント2：
    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よ

演習4（演算）

- 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)を作成し提出せよ

演習5（for文）

- 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)を作成しなさい

演習6（早く終わった人）

- ディレクトリ内の拡張子が.txtであるファイルをすべて「隠しファイル」にするシェルスクリプトを作成せよ
  - 隠しファイル：ファイル名の先頭にドット(.)を付与したファイル
  - 隠しファイルにするとlsコマンドでは基本的に表示されない
    （ls –aなど、-aオプションを付けると隠しファイルも表示される）
  - * この演習は間違えると予期せぬファイルが隠しファイルになる可能性がある。必ずGitでバージョン管理しておくこと

次回

- C言語からOSの機能を呼び出す「システムコール」という仕組みを学びます
- 次々回は試験前の演習回になります。