A. BASH

REFERENCES

- "Advanced Bash-Scripting Guide" [web]

EDITOR

- For Unbuntu/Linux novice, recommend `gedit`. You can type `gedit filename` in Ubuntu Linux shell to launch the editor.
- For advanced user, recommend `vim`

FILE MANAGEMENT

- touch, mv, rm
- chmod: permission bits
  - umask
- ln: symbolic/hard link

SHELL PROGRAMMING (FEB. W3)

INTRODUCTION

- Why learn shell programming?
  - automate administrative tasks, save your efforts!
  - e.g. automatic software update, file backup, resource monitoring
- Script: tie shell commands in a file
- Execute script `script.sh`:
  - `./script.sh`
  - `source script.sh`

BASICS: SHELL LANGUAGE

- `#!` sha-bang is a two-byte magic number that designates a file t
  - basically says it's an executable shell script
- Language syntax: if/else, variable
- demo:

1. `#!/bin/bash echo 'hello world';`
2. `#!/bin/bash a=1;b=2;a=$a+$b;echo $a;`
3. `#!/bin/bash a=1;b=2; a=$((a+b));echo $a;`
4. `if [ $a -gt $b ];then echo 'a larger than b`

- exercise:
  1. * try `a=1;b=2;c=$a;a=$b;b=$c;echo $a,$b;`, and put the output to the **blackboard**.
  2. write a script to initialize variables `a`, `b`, `c` and print their sum.
  3. write a script to swap the names of two files, file1 and file2. For example if input file1 contains Alice and file2 contains Bob at the beginning, after the execution, file1 should contain Bob and file2 should contain Alice.

PASSING ARGUMENTS

- demo:
  - `#!/bin/bash echo $1; echo $2; echo $#;`
- exercise:
  1. `#!/bin/bash a=$1; b=$2; echo $((a*b));;` try this script and tell what it does?
  2. write a script to get 3 integers from the command-line and prints their product.
     - what happens if you do not pass the 3 required integers when executing the bash script?

COMMENTING

- # is used to comment in bash

GREP & FIND (FEB. W4)

INTRODUCTION

- A classic matching problem:
  - takes as input a string and "pattern", outputs a binary decision.
  - `substring_match(al*ce,"alice bob")=1`
- Format of the pattern: **regular expression**.
- Relevance to Linux shell: `grep`, `find`, search in `vim`

REGULAR EXPRESSION & GREP

1. asterisk `*`
   - `match(1133*,113)=1`
   - demo: `echo 113 | grep 1133*`
2. dot `.`
   - `match(13.,13)=0`
   - `match(13.,134)=1`
3. Brackets `[...]`: enclose a set of characters
   - `match(1[345],13)=1`, `match(1[345],15)=1`, `match(1[345],18)=0`
   - `match(1[3-5],14)=1`
   - `match(1[^3-5],14)=0`,`match(1[^3-5],18)=1`

3. caret `^`: beginning of a line
4. dollar sign `$`: end of a line
    - `^$` matches blank lines.

GREP

- Demos
  1. `grep hello hello.c`
  2. `grep -r hello .`
  3. `grep -i HELLO hello.c`

- Exercise:
  2. Given a file `file1` with `hello too`, try following commands, and report the result
     - `grep ^hello file1`
     - `grep hello$ file1`
     - `grep t[wo]o file1`
     - `grep ^[A-Z] file1`
  3. Write a command to find all lines of all files under current directory recursively that contain a single word "hello".

FIND

- demo:
  1. `find`
  2. `find . -name "*.c"`
  3. `find / -maxdepth 1 -type d`
- exercise:
  4. write a command to find all the files with name starting with `fil` under the current directory.