CE 350

Lecture 3

UNIX/Linux Shells

by İlker Korkmaz and Kaya Oğuz

To Be Covered: UNIX/Linux Shells

- UNIX Shells
- Linux Shells
- A Little Background:
  - Processes
  - Environment and Inheritance
- Shells
  - Bourne
  - C and TC
  - Korn
  - Bash
- A Quick Start to Shell Programming
  - Some in class examples as LAB#1 work

UNIX Shells

- The shells associated with AT&T UNIX:
  - Bourne
  - Korn
- The shells associated with Berkeley UNIX:
  - C (csh)
  - TC (tcsh) is an enhanced version of the Berkeley C shell.

Linux Shell

- The shell associated with Linux:
  - Bourne Again (bash)
- Various shells may be used in Linux systems.

Responsibilities of the Shell

- The followings are the main responsibilities of the shell:
  - reading input and parsing the command line
  - evaluating special characters (wildcards)
  - setting up pipes, redirection and background processing
  - handling signals
  - setting up programs for execution

System Calls

- If a command is an executable program on disk, shell arranges for execution of the program. This is done with calls to the kernel (system calls).
- A system call is a request for kernel services.
- The shell can spawn (fork) other processes through system calls.

A Little Background on Processes

- In order to understand the relation with the process concept and the shell, the context of Section 5 of Chapter 1 of the textbook shall be dissected.
  - Read the pages 14–18 of the textbook in order to review the concepts of
    - fork
    - wait
    - exec
    - exit

A Little Background on the Environment and Inheritance

- Read Section 6 of Chapter 1 of the textbook to understand the concepts of
  - variables
  - redirection (<, >, >>)
  - pipe (|)
- about redirection:
  - stdin → 0
  - stdout → 1
  - stderr → 2
- about pipe:
  - Dissect the following:
  - who | wc
- More about environment variables:
  - http://en.wikipedia.org/wiki/Environment_variable

More about variables

. The shell can define two types of variables: local and environment.
.Local variables are private to the shell in which they are created and not passed to any child processes.
. Some of the environment variables are inherited from the /bin/login program, some others are created later in user initialization files, in scripts, or at the command line.
.An environment variable set in the child shell is not passed back to the parent shell.

.TO DO:

- –Try the followings:
  - .printenv
  - .which printenv

The Environment and Inheritance

- When you log on, the shell starts up and inherits a number of variables, I/O streams, and process characteristics from its parent started it (that is /bin/login).
- If another shell is spawned (forked) from the login or parent shell, that new child shell (subshell) will inherit the environment of its parent.
- The environment consists of process permissions, the working directory, the file creation mask, special variables, open files, and signals.

The Environment and Inheritance (2)

- What has been done before the shell begins?
  - init is the first process with PID=1.
  - init initializes the system and then starts other processes to open terminal lines and stdin-stdout-stderr. After terminal opens, /bin/login program runs. It checks the given account name with the first field in “passwd” file. Then “crypt” checks the password. After verification, login program sets the related environment variables, HOME, SHELL, PATH, ...
  - When login has finished it will execute the last entry in passwd file, which is probably /bin/bash

Executing Commands from Scripts (a little interactivity)

```
#!/bin/csh (or #!/bin/bash)
echo hello:)
ls
who
date
pwd
echo bye:(
```

- Open a new file and type above commands one per line.
- Make the file executable.
- Execute the file, which is called a *script*.

Shells

- Basic shells are:
  - Bourne
  - C and TC
  - Korn
  - Bash
- Comparison between these shells according to their features is available on page 1103 in Appendix-B of the textbook.

Shell Programming QuickStart

. The C shell and TC shell is based on the C programming language.
. The Bourne shell is based on Algol.
. The Bash and Korn shells are based on Bourne shell and also C shell together.

An Overview of Bash Shell Syntax and Constructs

- Examine the brief info in Table 2.4
- An overview on bash syntax:
  - shbang: **#!**
  - comment sign: **#**
  - metacharacters (wildcards): ***** , **?** , ...
  - displaying output: **echo** “Hello Linux World”
  - local variables, global variables
  - dollar sign to extract the value from a variable: **$**PATH
  - reading user input: **read**
  - command substitution: **$(** command **)** , **`** command **`**
  - arrays, operators, integer arithmetic, positional parameters to store arguments, conditional statements, loops, functions ...

LAB#1:

- The “shbang” line of all scripts consists of a #! followed by the full pathname to the shell.
- C or TC scripts:
  - #!/bin/csh or #!/bin/tcsh
- Bourne scripts:
  - #!/bin/sh
- Korn scripts:
  - #!/bin/ksh
- Bash scripts:
  - #!/bin/bash

- LAB#1 WORK: SAMPLE SCRIPTS TO DISSECT ...