Study Questions: Set No. 2
Introduction to UNIX
Sunday September 29, 2013

Covering:
Topic 06: *Unix Processes and Job Control*
Topic 07: *Unix Shell Environments*

1. What is the Unix process?
2. What are the three main states that any living process may have?
3. Under which condition does a Unix process enter a blocked state?
4. In Unix, what is the difference between a process ready state and blocked state?
5. What is the Unix PID?
6. What is the Unix PPID?
7. In a Unix operating system, scheduling services are performed by a program called ***scheduler***. What are the main tasks that a ***schedule*** perform?
8. List five of the Unix system calls that coordinate processes. Describe the main function of each of them.
9. What are the main differences between internal and external Unix commands?
10. Explain what will happen when executing a Unix external command.
11. In Unix, what is the relation between child and parent processes?
12. Explain in details what will happen when a process executes the fork system call.
13. Explain in details what will happen when a process executes the exec system call.
14. What is the Unix Zombie state?
15. When does a Unix process enter a Zombie state?
16. What is the Unix job?
17. In Unix systems, what is the only living ancestor of all other processes in the system? Explain why.
18. In Unix, what is an orphan process?
19. In Unix, when does a process become an orphan?
20. In Unix, who does adopt orphan processes?
21. In Unix, why orphan processes need to be adopted?
22. In Unix systems, what will happen if a parent process unexpectedly died before its child finishing its process?
23. In Unix systems, what are the main differences between a foreground and a background executions of a program?
24. List five of the Unix signals that are used with the kill command. Describe the main function of each of them.
25. How similar the Unix stop and kill commands?
26. How similar the Unix `^z` and stop command?
27. How similar the Unix `^z` and kill command?
28. How similar the Unix `^c` and kill command?
29. How do you run a Unix job in the background?
30. How do you run a Unix job in the foreground?
31. How do you move a Unix job from the foreground to the background?

32. How do you move a Unix job from the background to the foreground?
33. In a C-shell, what will happen if a background task
    a. sends output to the standard output, or standard error, and you have not redirect them?
    b. requests input from the standard input, and you have not redirected the standard input?
34. In a Bourne-shell, what will happen if a background task
    a. sends output to the standard output, or standard error, and have not redirect them?
    b. requests input from the standard input, and you have not redirected the standard input?
35. Can you start typing the next command before the previous command completed its execution? If yes, how?
36. How do you display information about the current Unix processes?
37. How do you display information about your current Unix jobs?
38. What is the difference between killing and stopping a Unix process?
39. How do you kill a Unix process?
40. You started a program named ABC but you realized that it is not behaving and it will take a long time to finish execution. What steps would you take to kill it? If ABC does not respond, what do you do?
41. How do you stop a Unix process?
42. How do you resume a stopped Unix process?
43. In Unix while a program is running, what does happen when you press **`^c`**?
44. In Unix a program is running, what does happen when you press **`^z`**?
45. In Unix, what is the daemon process?
46. Does Unix daemons work in the foreground or in the background?
47. In Unix, who is the parent of all daemon processes?
48. List eight Unix daemon processes. Describe the main function of each of them.
49. When you give a shell more than one command or when you write a shell script, you must separate commands from each other. List 5 command separators in Unix. List the characteristics of each one of them.
50. Assuming that you are running a Bourne-shell and the **abc** file does not exist, what is the output of the following Unix commands: **cd /tmp; pwd > abc ; ln abc ABC; rm abc; cat ABC**
51. The following is a simple Unix Bourne shell script (program). This program displays the shell script filename (`$0`), the iteration number (`$i`) followed by a colon (`:`), and the date/time of displaying this information. The program will do so three times--one second a part (`sleep 1`). At the end, it will display a message saying that the program finished execution.

```
#!/bin/sh
i=1
while [ $i -le 3 ]; do
  echo $0 $i ":"   `date`
  sleep 1
  i=`expr $i + 1`
done
echo $0 ": finished execution."
```

    a. Type in the above script in a file called prog-A
    b. Copy prog-A to prog-B and copy prog-A to prog-C
    c. Change the permission of prog-A, prog-B, and prog-C file to give execution permission to the user owner.
    d. What will be the output if you execute the following commands (below)?
    e. Draw a time chart to show when, approximately, each program will start and end.
    f. How do you stop each of these programs during its execution?

```
      i. Prog-A;  prog-B;  prog-C
     ii. Prog-A;  prog-B;  prog-C;
    iii. Prog-A;  prog-B   prog-C
     iv. Prog-A;  prog-B   prog-C;
      v. Prog-A   prog-B;  prog-C
     vi. Prog-A   prog-B;  prog-C;
    vii. prog-A   prog-B   prog-C
   viii. prog-A   prog-B   prog-C;
     ix. prog-A&  prog-B;  prog-C
      x. prog-A&  prog-B   prog-C
     xi. prog-A&  prog-B&  prog-C
    xii. prog-A;  prog-B&  prog-C
   xiii. prog-A   prog-B&  prog-C
    xiv. prog-A   prog-B&  prog-C&
     xv. prog-A&  prog-B&  prog-C&
    xvi. prog-A&  prog-B   prog-C&
   xvii. prog-A; (prog-B;  prog-C)
  xviii. prog-A; (prog-B   prog-C)
    xix. prog-A; (prog-B&  prog-C)
     xx. prog-A& (prog-B;  prog-C)
    xxi. prog-A& (prog-B   prog-C)
   xxii. prog-A& (prog-B&  prog-C)
  xxiii. (prog-A&  prog-B); prog-C
   xxiv. (prog-A;  prog-B); prog-C
    xxv. (prog-A   prog-B); prog-C
   xxvi. (prog-A&  prog-B)& prog-C
  xxvii. (prog-A;  prog-B)& prog-C
 xxviii. (prog-A   prog-B)& prog-C
   xxix. Prog-A | prog-B;  prog-C
    xxx. Prog-A | tee prog-B;  prog-C
   xxxi. Prog-A | prog-B&  prog-C
  xxxii. Prog-A | tee prog-B&  prog-C
 xxxiii. Prog-A | prog-B   prog-C
  xxxiv. Prog-A | tee prog-B   prog-C
   xxxv. Prog-A;  prog-B | prog-C
  xxxvi. Prog-A;  prog-B | tee prog-C
 xxxvii. prog-A&  prog-B | prog-C
xxxviii. prog-A&  prog-B | tee prog-C
  xxxix. prog-A   prog-B | prog-C
     xl. prog-A   prog-B | tee prog-C
    xli. prog-A | prog-B | prog-C
   xlii. prog-A | tee prog-B | tee prog-C
  xliii. Prog-A | prog-B;  prog-C&
```

```
   xliv.  Prog-A | tee prog-B;  prog-C&
    xlv.  Prog-A | prog-B&  prog-C&
   xlvi.  Prog-A | tee prog-B&  prog-C&
  xlvii.  Prog-A | prog-B   prog-C&
 xlviii.  Prog-A | tee prog-B   prog-C&
   xlix.  Prog-A;  prog-B | prog-C&
      l.  Prog-A;  prog-B | tee prog-C&
     li.  prog-A&  prog-B | prog-C&
    lii.  prog-A&  prog-B | tee prog-C&
   liii.  prog-A   prog-B | prog-C&
    liv.  prog-A   prog-B | tee prog-C&
     lv.  prog-A | tee prog-B | tee prog-C&
    lvi.  prog-A | prog-B | prog-C&
   lvii.  prog-A | tee prog-B | tee prog-C&
  lviii.  prog-A |(prog-B;  prog-C)
    lix.  prog-A | tee (prog-B;  prog-C)
     lx.  prog-A |(prog-B&  prog-C)
    lxi.  prog-A | tee (prog-B&  prog-C)
   lxii.  prog-A |(prog-B   prog-C)
  lxiii.  prog-A | tee (prog-B   prog-C)
   lxiv.  prog-A; (prog-B | prog-C)
    lxv.  prog-A; (prog-B | tee prog-C)
   lxvi.  prog-A& (prog-B | prog-C)
  lxvii.  prog-A& (prog-B | tee prog-C)
 lxviii.  prog-A  (prog-B | prog-C)
   lxix.  prog-A  (prog-B | tee prog-C)
    lxx.  prog-A |(prog-B | prog-C)
   lxxi.  prog-A | tee (prog-B | tee prog-C)
  lxxii.  (prog-A; prog-B)| prog-C
 lxxiii.  (prog-A; prog-B)| tee prog-C
  lxxiv.  (prog-A& prog-B)| prog-C
   lxxv.  (prog-A& prog-B)| tee prog-C
  lxxvi.  (prog-A  prog-B)| prog-C
 lxxvii.  (prog-A  prog-B)| tee prog-C
lxxviii.  (prog-A| prog-B); prog-C
  lxxix.  (prog-A| tee prog-B); prog-C
   lxxx.  (prog-A| prog-B)& prog-C
  lxxxi.  (prog-A| tee prog-B)& prog-C
 lxxxii.  (prog-A| prog-B)  prog-C
lxxxiii.  (prog-A| tee prog-B)  prog-C
 lxxxiv.  (prog-A| prog-B)| prog-C
  lxxxv.  (prog-A| tee prog-B)| tee prog-C
```

52. Predict the output of the following commands:
```
cd ~
pwd
tcsh
cd /
pwd
exit
pwd
```
53. In `tcsh` shell, how do you create a regular variable?
54. In `sh` shell, how do you create a regular variable?
55. In `tcsh` shell, how do you change a value of a regular variable?
56. In `sh` shell, how do you change a value of a regular variable?
57. In `tcsh` shell, how do you delete a regular variable?
58. In `sh` shell, how do you delete a regular variable?
59. In `tcsh` shell, how do you create an environment variable?
60. In `sh` shell, how do you create an environment variable?
61. In `tcsh` shell, how do you change a value of an environment variable?
62. In `sh` shell, how do you change a value of an environment variable?
63. In `tcsh` shell, how do you delete an environment variable?
64. In `sh` shell, how do you delete an environment variable?
65. What is the difference between regular shell variables and environment shell variables?
66. In `tcsh` shell, how do you display all current environment variables?
67. In `tcsh` shell, how do you display all current regular variables?
68. In `sh` shell, how do you display all current environment variables?
69. In `sh` shell, how do you display all current regular and environment variables?
70. Is there a command under `sh` shell that can display current environment variables alone?
71. In `tcsh` shell, create an *alias* named `rm` that always deletes files recursively.
72. In `tcsh` shell, create an *alias* named `ls` that always display files with long format.
73. In `tcsh` shell, how would you configure the Unix history to store the last 200 commands?
74. In `tcsh` shell, how would you configure the Unix history to store the last 100 commands between two consecutive sessions?