Arab Academy for Science, Technology, and Maritime Transport

College of Computing and Information Technology
Graduate Diploma - CS621 Operating Systems

Fall 2012

Tutorial Assignment

Instructor: Dr. Ayman Abdel-Hamid
TA: Eng. Ahmed Elbarbary

Due date: 13/1/2012

Please solve all the following questions:

1. Name some tasks (at least two) performed by the kernel. (2 points)
2. Can you have the same login name more than once in the "who" output? What would cause that? (1 point)
3. What is the different between the two commands "whoami" and "who am i"? (1 point)
4. How do you list the attributes (i.e., the permissions, the owner, the size, etc.) of the current directory (and not the contents of the directory)? (1 point)
5. A file, file1, has been created with default permissions 664. Will the file permissions will be the same if these commands are applied on the file in its original state?
    a. chmod +x file1
    b. chmod 755 file1

    (1 point)
6. How will you find out the total disk usage of the current directory tree (i.e., the size of the current directory and recursively all contents)? (1 point)
7. What will the following command do?
    $ cat namesfile | wc -l > number (1 point)
8. What is the PID of the init daemon? (1 point)
9. What is the default nice value if you typed the ps -l command? (1 point)
10. What is the command to view all daemons and PID information but only show daemons with the word bash in them? (1 point)
11. How do you find out the complete command line of a running process by a user named "student"? (1 point)
12. What is the difference between a process run with & and one run with nohup? (1 point)
13. What is the command to move a background process to the foreground? And what is the command to send a foreground process to the background? (1 point)
14. Show the output of that program and explain why the last line will print a parent's pid of value 1 instead of the actual parent's pid? Modify the program to solve that problem.

```
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
# include <unistd.h>

int main ()  {

pid_t child;

printf ("MAIN program process' pid: %d.\n", getpid () );
child = fork () ;

if (child==-1) {
   printf ("Error");
   exit (0);
   }
else {
   if (child!==0) {
      printf ("PARENT process running, pid is: %d.\n", getpid() );
      printf ("PARENT process running, son's pid is: %d.\n", child );
      }
```

```
   else {
      printf ("CHILD process running, pid is: %d.\n", getpid() );
      printf ("CHILD process running, parent's pid is: %d.\n", getppid() );
      }
   }

return (0);
}
```

(4 points)

15. How many lines of "Hello World" will be displayed when this piece of code is run

```
int main()
{
        fork();
        fork();
        fork();
        printf("Hello World\n");
}
```

Explain you answer showing the family tree between the generated processes.

(3 points)

16. Modify the following program so that a Zombie Process will appear and last for only 10 seconds

```
#include<stdio.h>

int main(){
  int pid = fork();
  int status;
  if (pid ==0 ) {
      printf("\n Hi\n");
  }
  else {
           for( ;;);
           waitpid(pid, &status,0);
  }
  return 0;
}
```

(3 points)

17. Describe what will be the output when the following program is run. If there is more than one possible output, describe all the possibilities.
    Here are some general important facts:

- The program is made up of two concurrent processes. You don't know in what order they will run, nor do you know when the dispatcher will switch between processes. However the dispatcher *will* eventually and continually switch back and forth between processes (i.e., no process will starve).
- The initialization code (setting the initial values for the shared variables), is completed *before* either of the two processes run.
- Both the variables (X and Y) are shared between the two processes.
- Every time a variable is referenced (appears in an expression), it is read from memory.
- Every time a variable is set (appears on the left-hand size of an assignment operator), it is written to memory.
- Reading and writing single words (ints) is atomic.

Initialization
`int X = 0;`
`int Y = 0;`	

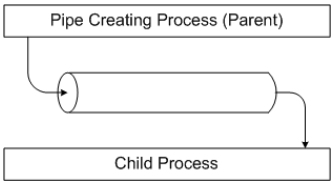
Process A `while (X == 0) {`	**Process B** `printf ("b");`

```
  // do nothing
}
printf ("a");
Y = 1;
Y = 0;
printf ("d");
Y = 1;
```

```
X = 1;
while (Y == 0) {
  // do nothing
}
printf ("c");
```

(3 points)

18. The following program shows a single flow of communicating from a parent process (producer) to a child process (Consumer) using a pipeline as show in the image.




```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>

int main(int argc, char **argv)
{
  int pfd[2];
  int nread;
  int pid;
  char name[80];

  if (pipe(pfd) == -1)
  {
    perror("pipe failed");
    exit(1);
  }

  if ((pid = fork()) < 0)
  {
    perror("fork failed");
    exit(2);
  }

  if (pid == 0)
  {
    /* child */
    close(pfd[1]);
    while ((nread =
         read(pfd[0], name, 80))
         != 0)
      printf("child read your name:%s\n", name);
    close(pfd[0]);
  } else {
    /* parent */
    close(pfd[0]);
    printf("Enter your name: " );
    gets(name);
    /* include null terminator in write */
    write(pfd[1], name,
          strlen(name)+1);
    close(pfd[1]);
  }
  exit(0);
}
```

Modify the program so that the child process will be the responsible for the writing to the pipeline; child process will be producer and parent will be consumer (3 points)