Programming Tools-II

Lab Assignment Week #2

1. Write a program that executes the "cp -p -i filename1 filename2" command. Call your executable myCp Details:
   (a) Look in the man pages to see what the -p -i options do when using cp
   (b) Create a file with the name of filename1. The contents of the file can be text of 500 words.
   (c) The call to your program will be made with the following command:
   % myCp filename1 filename2
   (d) Your code will check for the correct number of arguments. If it is not correct, then an error message will be produced and the code will exit.
   (e) Your code will fork()
   (f) The child will use the execl to call cp and use the filename1 and filename2 passed as arguments on the command line
   (g) The parent will wait for the child to finish
   (h) Your program will also print from the child process:
   • The process id
   • The parent id
   • The process group id and
   (i) print from the parent process:
   • the process id
   • the parent id
   • the process group id
   (j) Comment out the execl call and add instead a call to execv. Add any necessary variables to do that.
2. Answer the following questions as comments in the code you wrote above:
   (a) If you try to print a message after the exec* call, does it print it? Why? Why not?
   (b) Who is the parent of your executable (myCp) program?
   (c) How would you change the code so that the child and parent "appear" to run concurrently (ie. at the same time)?
   (d) What does -p -i mean for the cp

```
Sample run:
[1]%myCp
error myCp: not enough arguments
```

```
[2]%myCp filename1error myCp: not enough arguments

[3]%myCp filename1 filename2
In the CHILD process Trying to Copy
Child Process ID: 2014, Parent ID: 2013, Process Group: 2013
In the PARENT process
Original Process ID: 2013, Parent Is: 1889, Process Group is:
2013

[4]%myCp filename1 filename2
In the CHILD process Trying to Copy
Child Process ID: 2014, Parent ID: 2013, Process Group: 2013
cp: overwrite `backup'? y
In the PARENT process
Original Process ID: 2013, Parent Is: 1889, Process Group is:
201
```

3. Write a C program to create a child process using fork() system call. Also use wait() system call in parent process such that it gets the return status of its child, return status must be printed on console.

   Hint: Use (i) WIFEXITED

   (ii) WEXITSTATUS

   (iii) WIFSIGNALED

   POSIX macros to print the exit status of child process.

3. Use fork-exec combination to create a child process. The child process invoke a new program “firefox” using “exec” while parent process continue to exectute ***ls*** command with command-line argument ***-l*** in current directory.