Cse660 **Lab 5. Assignment** February 25, 2011.

Due date: Friday, March 4, 2011. **Part C is bonus!!**

Purpose: Pipe message exchange mechanisms among Unix processes.

<u>Assignment Part A.</u>: Write C (or C++) program with a source code in the file PipeRW.c, that works as follows:

1. You run a program PipeRW as a process, call it Process A. Process A creates a pipe and two child processes, call them Process B and Process C. **Those two processes will be executing only its parent's code, i.e. there is no exec system call in the code of Process A.**

2. Process B writes into the pipe the following character strings, one in each iteration: "001aaa", "002aaa", "003aaa", "004aaa", ... "009aaa", "010aaa", "011aaa", "012aaa", "013aaa", ... "019aaa", "020aaa", "021aaa",..., "030aaa", ..., "099aaa", "100aaa", ... , "291aaa", "292aaa", "293aaa" ,..., "300aaa". When done, i.e. after 300 iterations, Process B terminates.

3. Process C places into the pipe the following items (one by one, i.e. 3 characters at the time): "Ax0", "Bx0", "Cx0", ... "Yx0", "Zx0", "Ax1", "Bx1", "Cx1", ... "Zx1", ... "Ax9", "Bx9", .., "Zx9". When done, i.e. after 260 iterations, Process C terminates.

4. Process A reads from the pipe 100 characters at a time and writes all characters read by each read system call on the terminal. After Processes B and C terminate and pipe is empty, Process A terminates.

After putting each 50 of its strings in the pipe Process B sleeps for 1 sec, Process C sleeps for 2 seconds after putting each 60 of its strings in the pipe, while Process A sleeps for 1 sec after each 30 reads.

In this assignment, only system calls you are allowed to use are fork, sleep, exit, read, write, close, and pipe, plus any means of writing on the terminal.

Submit your source code files using command:

```
submit c660aa lab5a PipeRW.c
```

Also, provide a hard copy of your source code (file: PipeRW.c), a hard copy of the output (printing) of your program from one of executions and your compilation command.

Assignment Part B.*:* Write C (or C++) programs with source code in files PipeC.c, PipeR.c, PipeW1.c, and PipeW2.c, that work as follows:

1. First you run a program PipeC in a process, call it Process A. Process A does following:
 - creates a pipe
 - creates child process, with code from file PipeW1; call it Process B,
 - creates child process, with code from file PipeW2; call it Process C,
 - creates child process, with code from file PipeR; call it Process D, and then Process A terminates.

2. Process B behaves as Process B in Part A.

3. Process C behaves as Process C in Part A.

4. Process D behaves as Process A in point 4 of Part A.

In this assignment, only system calls you are allowed to use are fork, exit, read, write, close, pipe, and any of exec system calls, and any means of writing on the terminal.

Submit your source code files using command:

```
      submit c660aa lab5b PipeC.c PipeR.c PipeW1.c PipeW2.c
```

Also, provide a hard copy of your source codes (files: PipeC.c, PipeR.c *and* PiperW.c), a hard copy of the output (printing) of your programs from one of executions and compilation commands.
===

Assignment Part C.*:* Write C (or C++) program in the file Pipe.c, that mimics Unix shell functioning working as follows:

1. You run a program Pipe in a process, call it Process A, and it reads as an input from a keyboard two strings of characters (using two reads from a keyboard) that should be names of any Unix commands (Command_1 and Command_2) such the following makes sense as a shall input:

```
shall_prompt> Command_1 | Command_2
```

E.g. if one string is *ps* and another string is *more*, corresponding shall input would be ps | more. Do not check for syntax errors in Unix commands since they should be entered syntactically correct. Allow for Unix commands to have parameters. E.g. Two strings can be ls -l and *fgrep '.c'*, thus corresponding shall input would be *ls - l | fgrep '.c'*.

Then, Process A creates:
- a pipe,
- a process, that will receive code from the file associated with the first read,
- a process, that will receive code from the file associated with the second read.

2. The first child should write its output into the pipe instead of to the standard output (terminal).

3. The second child should read its input from the pipe instead of from the standard input (keyboard)

Process A waits for both of its child processes to terminate, and then it expects next input in the same format as defined in step 1. Process A terminates when it reads character @.

In this assignment, only system calls you are allowed to use are fork, exit, wait, pipe, dup2, and any of exec system calls, and any means of reading input from keyboard and writing on the terminal.

Submit your source code files using command:

```
submit c660aa  lab5c Pipe.c
```

Also, provide a hard copy of your source code (file: Pipe.c), a hard copy of the output (printing) of your program from one of executions and your compilation command.

<u>dup2 system call is needed for Part C</u>

dup2 system call allocates an entry in the process's open file pointer array, and initializes it to point to the existing open file structure (in the open file table). It is defined as:

```
#include <unistd.h>
```

int *dup2*(int *fileid*, int *copyfileid*)

Return: *copyfileid if OK*, -1 on error

copyfileid specifies the duplicated open file identifier (index) in PCB, while the index *fileid* specifies the existing open file identifier. If *copyfileid* is already in use, it is first released, as if *close(copyfileid)* had been performed.