McGill University
COMP 310/ECSE 427
Dr. Rola Harmouche

Assignment #1 **Winter 2018**
Due Date: 23:59 Hrs Friday, 2nd Feb 2018
Drafted By: Aakash Nandi and Dr. Rola Harmouche

1 Mini Programming Questions

1) Redirection [5 marks]

Given the following C program that prints data on screen, modify the program in order to redirect stdout to a file named **redirect_out.txt**, perform redirection then restore stdout (hint: use dup, dup2). In particular, the first and third printf statements should be written to standard out, and the second line should be written to file. Please save your code in a file named **a1_redirect.c**.

```
#include<stdio.h>
#include<unistd.h>
int main()
        {
        printf("First : Print to stdout\n");
        printf("Second : Print to redirect_out.txt\n");
        printf("Third : Print to stdout\n");
        return;
        }
```

2) Inter Process Communication [10 marks]

The following program serves as an example for inter process communication.Given below C program needs to be modified so that it execute an **ls** command in the child process, and prints the output of the command in the parent process.Please save your code in a file named **a1_command_piping.c**.(Hint: Use pipe,fork and execvp)

```
#include<stdio.h>
#include<unistd.h>
int main()
        {
        if(fork()==0)
                {
                //Child : execute ls using execvp
                }
         else
                {
                //Parent : print output from ls here
                }
        return 0;
        }
```

2 Emulating Shell Interface

1) Requirements

In this part of the assignment you are required to write a C program (using the skeleton provided) that emulates a terminal. The actual terminal has a provision which allows a command to run in the background. This is done by appending & to the command. However not all commands can be executed in background. Each command below is specified with information flags about this [F- foreground, B- background] . Now since the terminal has several commands, you are expected to implement a subset of commands which are mentioned below:

1. **Execution of built-in commands** which include cd[F],wc[F],jobs[F],exit[F].To get a better understanding of built-in commands and its specification please refer to corresponding section in Additional Information section.

2. **Execution of executable commands**. There are several commands that exits as executable files in **bin** directory.These include date,who,ls etc and can be run using execvp() function. Note that every executable command is [BF].

3. **Nice prefix for commands** Any command that is being run in foreground (this include commands that are BF and F) can be made to wait for background commands to finish first and then start its own execution by prefixing it with "nice".

4. **Output Redirection** is the feature that allows you to place the results of a command execution in a file provided (as filename) by the user.For ex. ls>filename.txt. Note that this feature is expected to work only with the executable commands.

2) Useful Information

1. **Processing input text** The skeleton code provides you a function that takes user input and breaks it into tokens of array of arguments. In all cases the first argument is the command and the second argument in the flag for that command.This function also sets a background variable to 1 if it encounters & and then removes & from the string.

   getcmd

2. **Background and Foreground** : This point applies only to executable commands. To enforce this feature you need to exploit the power of fork. By fork ,create a child process and in the child process you could have your commands run (which are suitable) using execvp. In case of the command being run as background the parent process need not wait the child process to terminate (look for suitable attribute for waitpid function) and in case of foreground the parent process has to wait for the child process to terminate.

3. **Built-in commands** : These commands are always run in the foreground and hence there is no need to fork.

   (a) **cd:** Requires you to call **chdir** system call.Takes destination directory as argument.If no argument is provided then change to home directory.

   (b) **wc:** Outputs either the number of lines (l flag .ie wc -l filename) or number of words (w flag .ie wc -w filename) present in the file provided by user.

(c) **exit:** Requires you to exit your terminal emulator.

(d) **jobs:** Requires you to print the details of the jobs that are running the background.Essentially refresh the linked list , traverse it and print [ID, PID, Command, Status, Spawn time]

4. **Executable Command** All the executable are of the type [BF] and require fork-execvp-waitpid calls.If you want to have a look at all the commands that can be run using execvp. Look into **bin directory** and you shall find executable files with command names.Refer to the tutorial on Assignment 1 to understand using fork-execvp-waitpid trio.

5. **Augmented Execution Time** All the commands in terminal execute very fast and you wouldn't be able to notice the background execution . In order to make it visible, the skeleton has a function which deliberately introduces random delay (0-15 seconds) before execution.You are requested to place this function calls at suitable location for the above purpose.

6. **Nice prefix** If a command is run with nice prefix that means before doing fork-execvp-waitpid, continuosly (at intervals of 1 sec) check the linked-list untill it becomes empty.Only then proceed with running the command. There exist a function in the skeleton that halts the parent process until the linked list is empty.Use this function cautiously.

7. **Output Redirection** You are expected to manipulate stdout stream (using dup or dup2) and then restore it, to dump the output of a command to a text file provided by the user.It should handle the case in which file does not exist(ie. create that file else append it to existing file).

3) Marking Scheme

Criteria	Weightage
A simple shell that runs: shows prompt, runs executable commands, goes to the next one, account for errors in inputs.	25
Built-in commands	30
Nice Command	15
Output Redirection	10
Code comments and documentation	5

You are provided with skeleton code and are expected to use it in order to complete your assignment. **Failure to do so will result in penalty. Any additional functionality that is not specified in this assignment statement will also result in penalty.**

A penalty of 5% shall be applied if the submission guidlines specified in the submission folder description are not followed.

————XX————