Operating System 111 Fall Homework 1

W.J. TSAI 蔡文錦 教授
TA 王菱君, 王麗婷, 余孟倫, 黃逸弘

Prework

Click here to apply for NYCU CSCC account

Download PuTTY

Using PuTTY to sign in to workstation.
Edit your C/C++ program at workstation.

Download FileZilla

Using FileZilla to download your file that save in the workstation.

PuTTY – Connect to Workstation

Click here to download PuTTY

Alternative binary files

The installer packages above will provide versions of all of these (except PuTTYtel and pterm), but you can download standalone binaries one by one if you prefer.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

putty.exe (the SSH and Telnet client itself)

64-bit x86:	putty.exe	(signature)
64-bit Arm:	putty.exe	(signature)
32-bit x86:	putty.exe	(signature)

Workstation Guide - NYCU CSCC

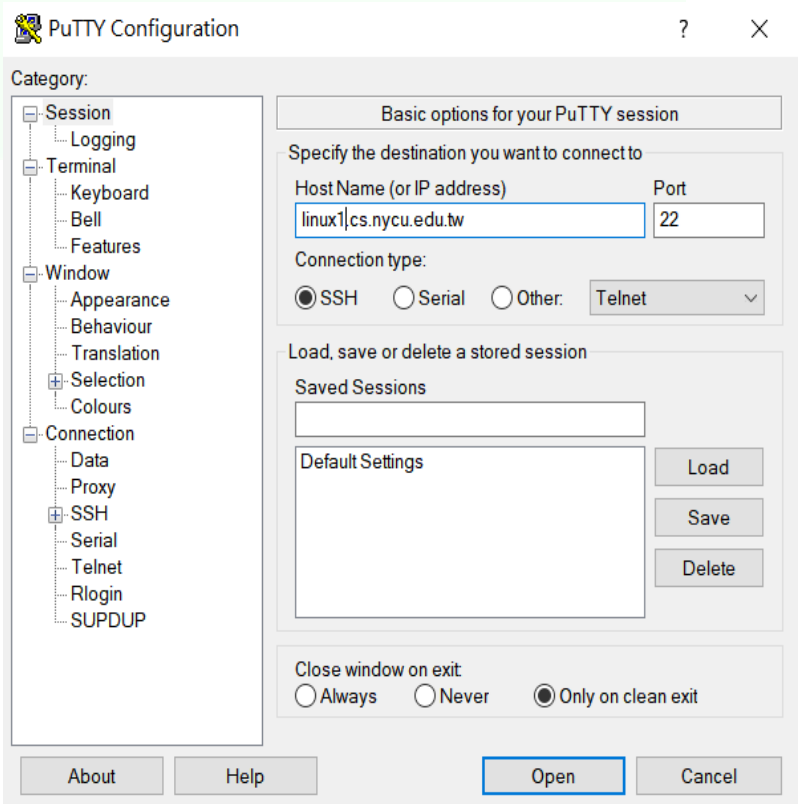
Login Settings

The default for SSH service is port 22

Host name

linux{1,2,3,4}.cs.nycu.edu.tw

bsd{1,2,3,4}.cs.nycu.edu.tw



FileZilla - Download File from Workstation

Click here to download FileZilla

Login Setting

- 協定：SFTP
- 主機：linux{1,2,3,4}.cs.nycu.edu.tw
- 連接埠：22
- 登入型態：一般
- 使用者：計中帳號
- 密碼：計中密碼

Homework 1

1-1

1-2

1-3

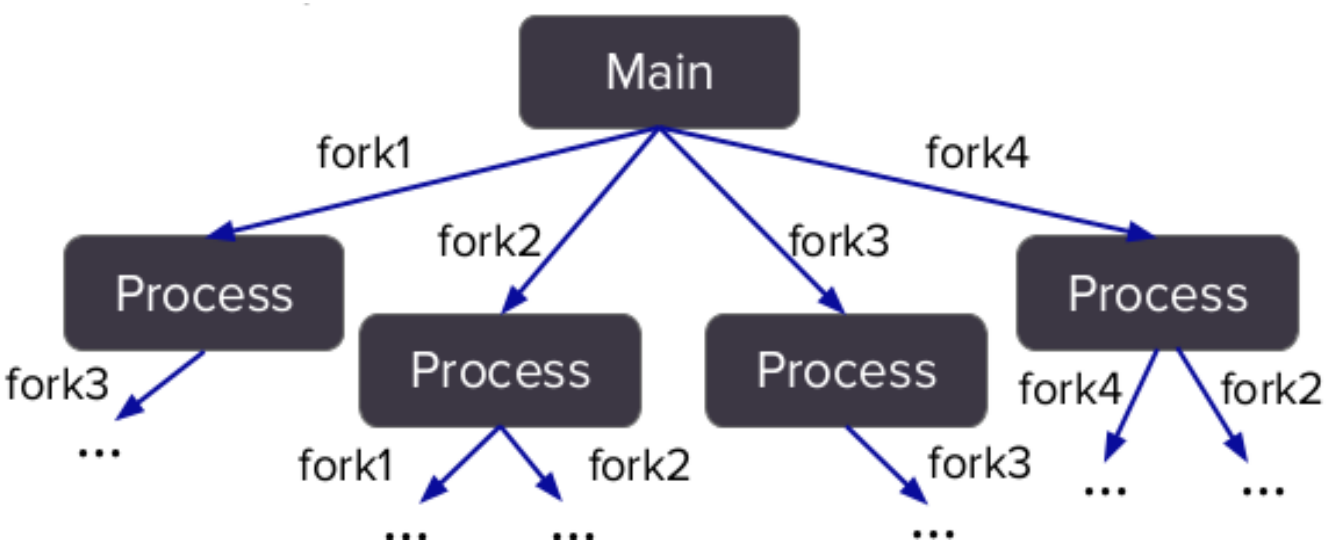
Submission and Grade

HW 1-1

Please draw the tree format according the code on the report (OS_report.pdf).

You need to clarify which fork (fork0, fork1, fork2, fork3 and fork4) the process been made by.

For example :




```c
#include <stdio.h>
#include <unistd.h>
#include<sys/wait.h>
#include<iostream>
using namespace std;
int main(){
    pid_t pid;
    pid = fork(); //fork0
    if(pid == 0){
        pid = fork(); //fork1
        if(pid > 0)
            wait(NULL);
        else if(pid == 0){
            pid = fork(); //fork2
            if(pid > 0)
                wait(NULL);
        }
    }
    else if(pid > 0){
        wait(NULL);
        pid = fork(); //fork3
        if(pid > 0)
            wait(NULL);
    }
    else{
        printf("Error!");
    }
    pid = fork(); //fork4
    if(pid > 0)
      wait(NULL);
    return 0;
}
```

Please follow this code to draw a picture

Hint

System Call

- getpid() : Get the process ID of the current process.
- getppid() : Get process ID of parent process.

Additions

- You can use above system calls to help you complete this part.
- Draw which fork (fork0, fork1, fork2, fork3, fork4) the process been made by after the code is executed.

HW 1-2

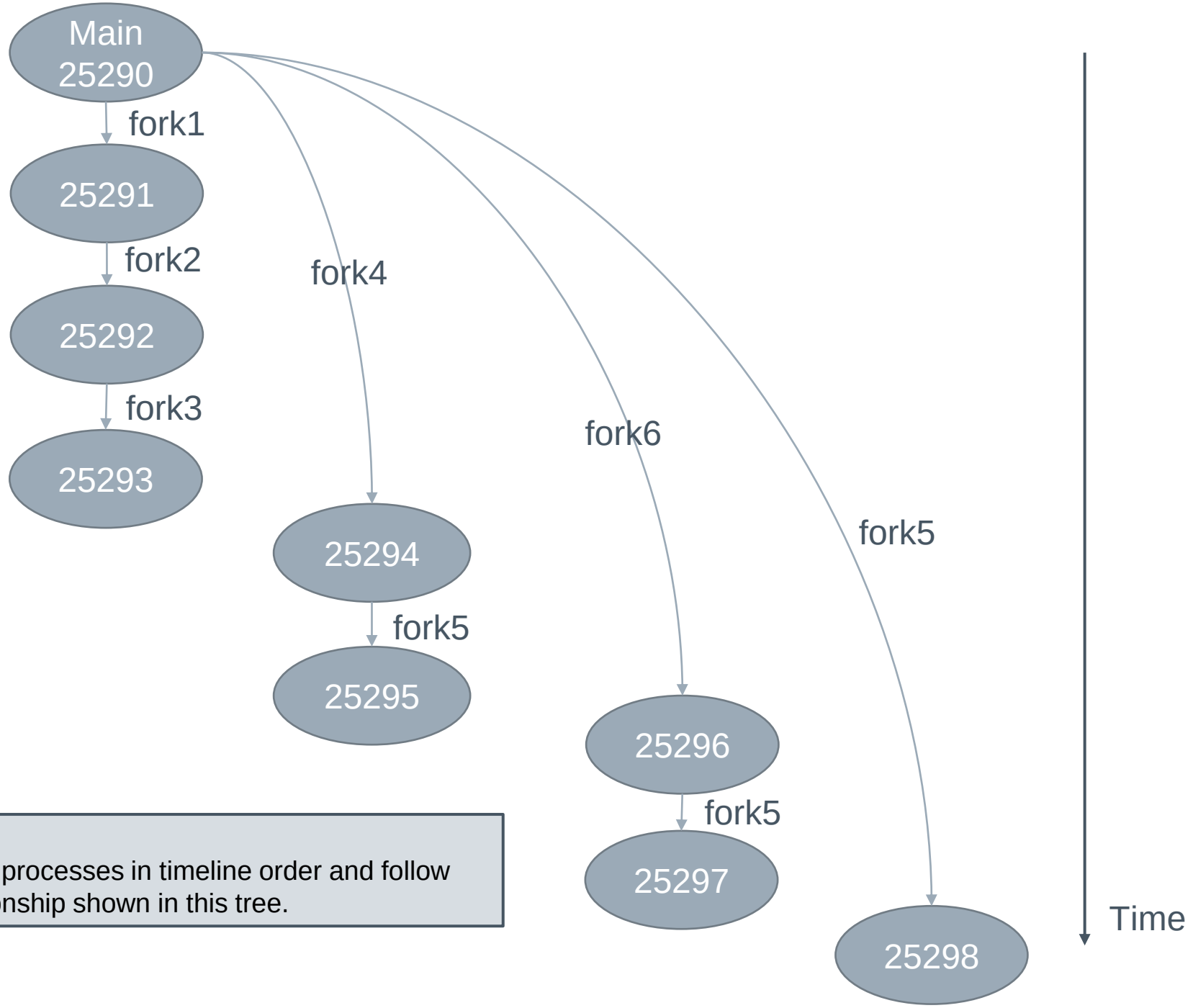
Write a program which uses fork() to produce the tree format in **next slide**.
Please note that **the format** and **the order of fork()** of your output should be the **same** as the following.

9 Children

```
Main Process ID : 25290
Fork 1. I'm the child 25291, my parent is 25290.
Fork 2. I'm the child 25292, my parent is 25291.
Fork 3. I'm the child 25293, my parent is 25292.
Fork 4. I'm the child 25294, my parent is 25290.
Fork 5. I'm the child 25295, my parent is 25294.
Fork 6. I'm the child 25296, my parent is 25290.
Fork 5. I'm the child 25297, my parent is 25296.
Fork 5. I'm the child 25298, my parent is 25290.
```

Hint :
To maintain the order of the fork(), parent processes have to **wait()** until child processes finish.

HW 1-2



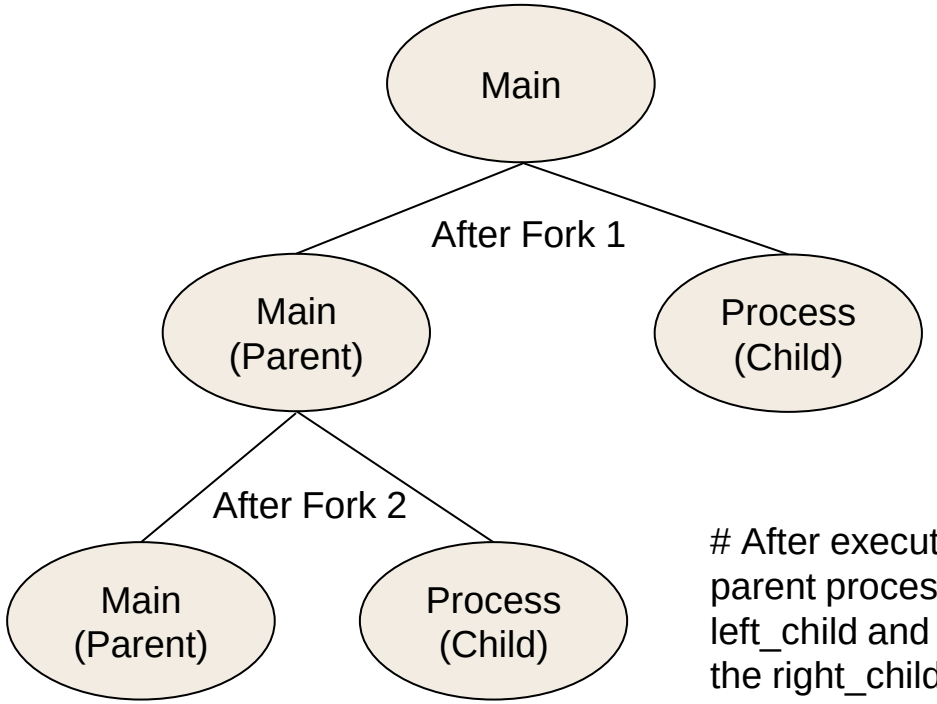
Note :
Your code must spawn processes in timeline order and follow the parent - child relationship shown in this tree.

HW 1-2 (Optional Reading)

We can also use an binary tree to record the result of processes after executing fork().

For example :

```
int main(){
    if(!fork()){//child process
        cout<<"Fork 1. I'm child. "
    }
    else{//parent process
        wait(NULL);
        if(fork()>0){
            wait(NULL);
        }
        else{
            cout<<"Fork 2. I'm child. "
        }
    }
    return 0;
}
```

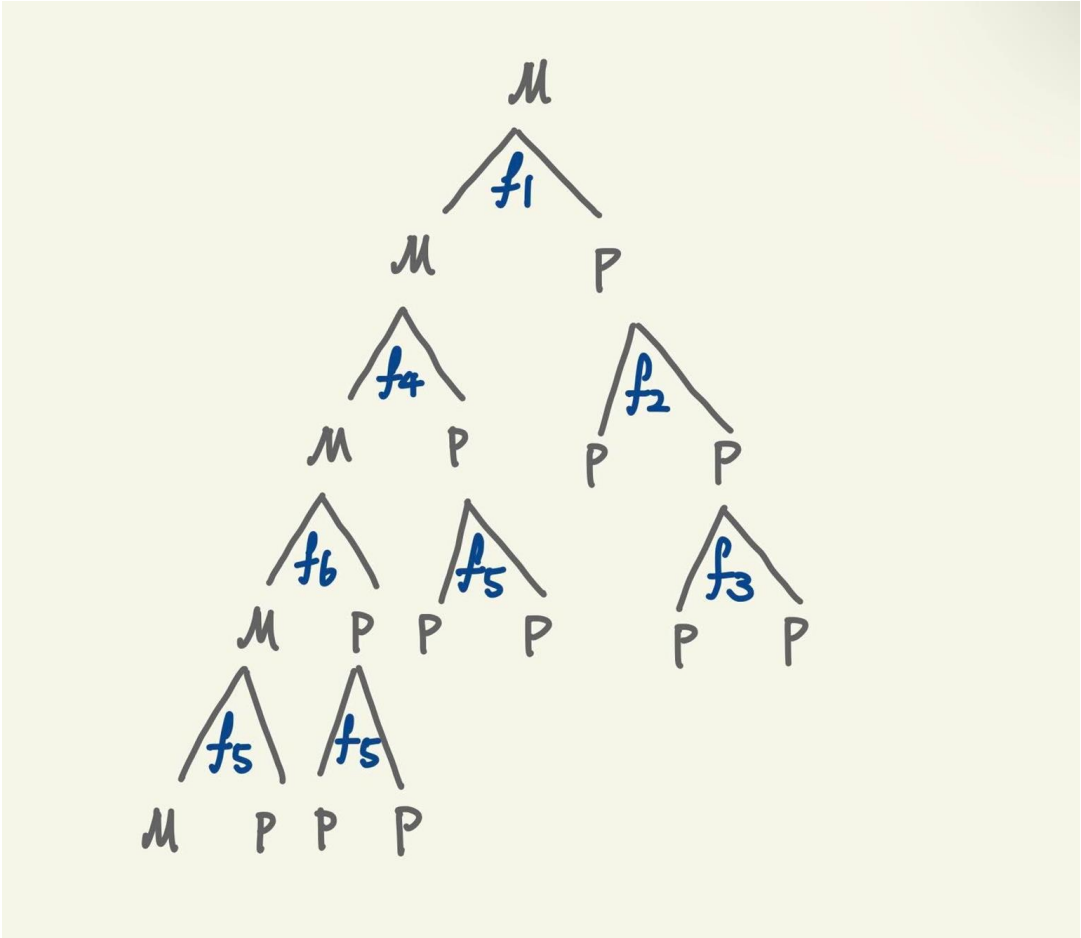


After executing fork(), let parent process to be the left_child and child process to be the right_child.

Binary tree notation of left hand side code

HW 1-2 (Optional Reading)

By using the binary tree notation, the result of HW1-2 will become :



HW 1-3

Finish “hw1_3.c” or “hw1_3.cpp” in order to design a C/C++ program to serve as a shell interface that accepts user commands then execute each command in a separate process.

Next page is an template for you.

Some important System Call :

- read (STDIN_FILENO, inputBuffer, MAX_LINE) : read command line
- fork() : Create child process
- execvp (char *command, char *params[]) : Execute system calls
- waitpid (pid)
- wait ()

HW 1-3 Example

Show the content of hello.txt

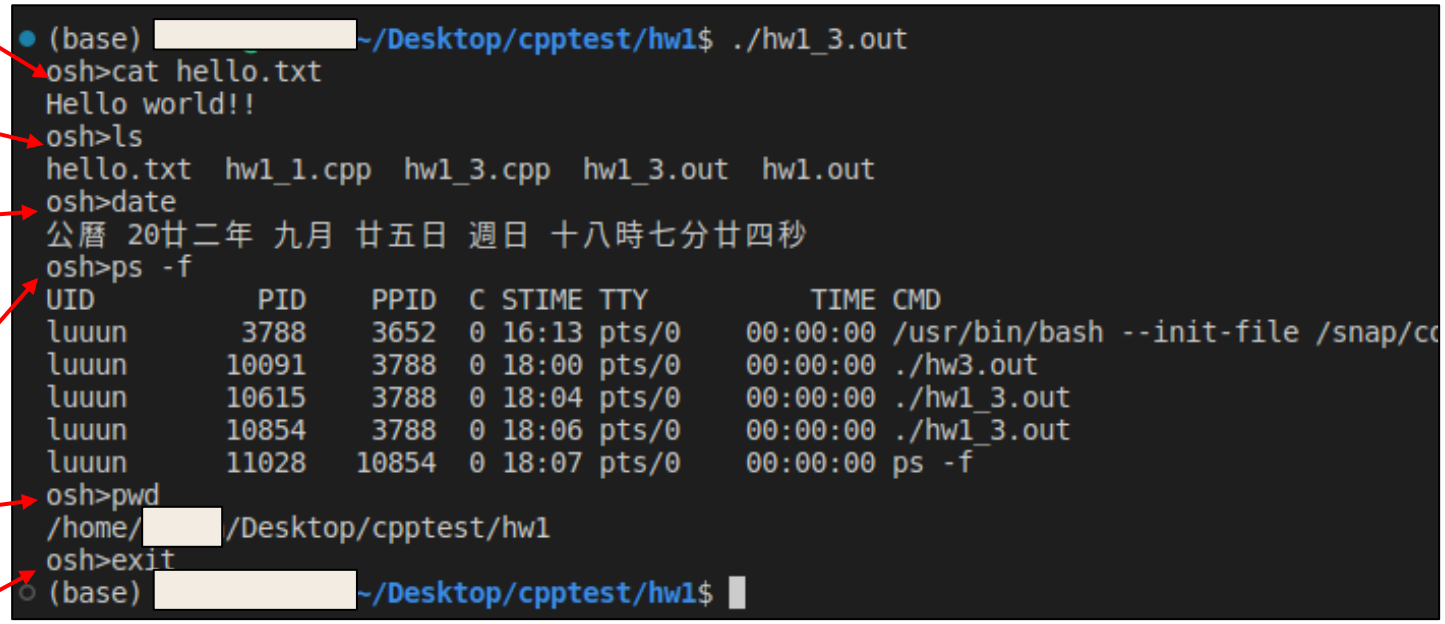
Show the file in the directory

Show the date

List all the processes in current shell

Show the path

Enter the “exit” to finish shell



HW 1-3

```
while(should_run){
    cout<<"osh>";
    fflush(stdout);
    n = read(STDIN_FILENO, buf ,80);
    for(int i=0;i<n;i++){
        if(buf[i]==' '||i==n-1){

            if(tmp.size()>0)
                arg.push_back(tmp);

            tmp = "";
        }
        else{
            tmp = tmp + buf[i];
        }
    }
    argc = arg.size();
    argv = new char*[argc+1];
    for(int i=0;i<argc;i++){
        argv[i]=new char[arg[i].size()];
        strcpy(argv[i], arg[i].c_str());
    }
    argv[argc] = new char;
    argv[argc] = NULL;
    /**
    * your code!
    * After reading user input, the step are:
    * (1) fork a child process using fork()
    * (2) the child process will invoke execvp()
    * (3) if command included &, parent will not invoke wait()
    */
    arg.clear();
    argv_t.clear();
    for(int i=0;i<argc;i++){
        delete [] argv[i];
    }
    delete argv;
}
return 0;
```

Here is the template for you

HW 1-3

You need

- Finish “hw1_3.c/hw1_3.cpp” as a shell interface.
- User can keep entering the command until he/she enters “exit”. (a command include the command itself and its parameters).
- Your shell needs to support following commands :

  cat, ls, date, ps(ps –f), pwd, exit

Submission and Grade

- Total score : 100pts.
- **Copy will get 0 point.**
- 1-1 score : Report Q1 30pts
- 1-2 score : Code 25pts / Report Q2 10pts
- 1-3 score : Code 25pts / Report Q3 10pts
- Incorrect output format will get -5 pts
- Report : Format is in OS_report.pdf.

Submission and Grade

- Use **only C/C++,** other language will get 0 point.
- Filename format please according to : **hw1_2.cc (or .cpp), hw1_3.c (or .cpp), OS_report.pdf**
- Incorrect filename format will get -5 pts
- Put **all** *.c (or *.cpp) files and *.pdf reports into same compressed file named **Student ID_hw1.zip**. (ex : 0000000_hw1.zip)

  Deadline : 2022/10/12 11:59 p.m
- Delayed submission will get -20% point a day.
- If you have any question, just send email to TAs by E3.