UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 122 Operating Systems

Practical 2

Due: 2017-09-04 @ 14:00 PM

August 14, 2017

PLAGIARISM POLICY

UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.ais.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then click the *Plagiarism* link). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

Instructions

In this module it is advised that you have a Virtual Machine(VM) to do all your practicals due to the nature of this course it is easy to mistakenly destroy your Operating System. For all practicals you must make use of a Linux VM.

Follow the Upload and Demo instructions carefully at the end of the practical.

- All written answers must be in either txt or pdf format no other formats will be marked.
- Only upload your source code, makefile, and supporting documents. Do not upload large, binary, or OS files. Keep your uploaded archive small.
- Upload your work in a tar or gz file before 14:00 on the 04th of September 2017.
- Bring your VM on a flash disk to your booked practical demonstration session in the week starting on the 05th of September 2017, so you may be marked.
- On non-demo days, there will still be teaching assistants available in the labs to help you.
- Make use of a Makefile for your program.
- You makefile **must** have these commands:
  - # make
  - # make run

Upload Instructions

As it would be impractical to upload your Virtual Machine to the CS website for the practicals, you are required to submit your code and answers to the written questions:

- Upload your work in one tar or gz file to the Practical 2 assignment slot on the COS 122 course website before 14:00 on Monday 04 September 2017. No late submissions will be accepted.
- You have to demo what you uploaded.
- All written answers must be in either txt or pdf format no other formats will be marked.
- Every file you submit should contain your name, surname, and student number at the top of the file in comments.
- Only upload your source code, makefile and supporting documents (written answers). Do not upload large binary, or OS files. Keep your uploaded archive small.

- **Failure to upload your work will result in 0 marks being awarded for your practical.**

Demo Instructions

At the demo, we will check your uploaded work. Only students who have uploaded on time will be allowed to demo. **You may only demo during the slot where you are booked**. You should be able to demo the practical within Virtual Box.

- This practical must be marked by a tutor or teaching assistant during your booked practical demonstration sessions.
- You have to be there at the start of the session until you get marked. If you come just before the session ends you will not get marked.
- You have a limited time to demo approximately 4 minutes. Please ensure that your VM is running before you get marked to speed up the process.
- If you are getting marked next and you are not ready to demo you will receive 0 marks for this practical.
- If your program doesn't compile due to syntax errors you will get 0 marks.
- If your program receives a runtime error you will lose marks.
- You have to be prepared to answer any question(s) the teaching assistant or tutor may ask you. You should not have to waste time looking for the answer.
- Make sure that you sign the attendance register for the session and ensure that the teaching assistant has entered your mark on the mark sheet correctly.
- **Failure to demo your work will result in 0 marks being awarded for this practical.**

Total: [22 Marks]

Task 1 - Processes [5 Marks]

Task 1(a) - 2 marks

In this task, you are required to write simple pipe commands.

(a) In this task, you are required to write a simple pipe command using **pgrep**. The command takes the name of a running process and provides the *process_ID* (PID).

(b) Now, write another pipe command to kill that particular process you identified in (**a**) above.

(c) Using **pgrep** and **ps**, write a command that can be used to show a full list of running processes of a given user using their *user_ID*.

(d) Using the *user_ID* in (**c**) above, write a pipe command that can be used to kill all running processes of that same user.
NOTE: You are advised to test the command in (d) using a VM otherwise running the command on a live machine will lead to all processes to be killed and you will be logged out of the system.

Task 1(b) - 3 marks

In the segment of this task, you are required to write a C++ program that captures and lists all the contents of the current directory in long list format by using a sub-process call. You must make use of a function to do this. You are not allowed to use **execvp** or friends.

- In your program, simply just pass the command as a string parameter to that function.
- Note that your code will likely not perform any error-checking, so make sure you test your commands output carefully else your executable will hang if bad output was received.
- Do enough error checking by validating your input and output at every stage.

Your output should look something like this.

```
user@user-desktop:~/Desktop/COS122/Linux$ ./a.out
LS: total 32
drwxrwxr-x 2 user user  4096 Aug 13 20:54 .
drwxrwxr-x 9 user user  4096 Aug  9 18:42 ..
-rwxrwxr-x 1 user user 14457 Aug 13 20:54 a.out
-rw-rw-r-- 1 user user   590 Aug  9 18:36 linux.cpp
-rw-rw-r-- 1 user user   474 Aug  9 18:33 userinput.cpp~

user@user-desktop:~/Desktop/COS122/Linux$
```

Task 2 - Theory on Threads [5 Marks]

Task 2(a) - 3 marks

In this task you are required to read up about pthreads (POSIX) and answer the following questions in a summarized form. You should be able to answer these questions in person during the demo session as well.

(a) What is POSIX?

(b) Why was it created?

(c) Mention a few operating system(s) and programming language(s) where POSIX is available. For instance Cygwin, Uwin, MinGW which are all classified under POSIX for Microsoft Windows.

Task 2(b) - 2 marks

Briefly answer the following questions.

(a) Discuss the difference between processes (done in practical 1) and threads (done in practical 2).

(b) What are the benefits and drawbacks of each approach?

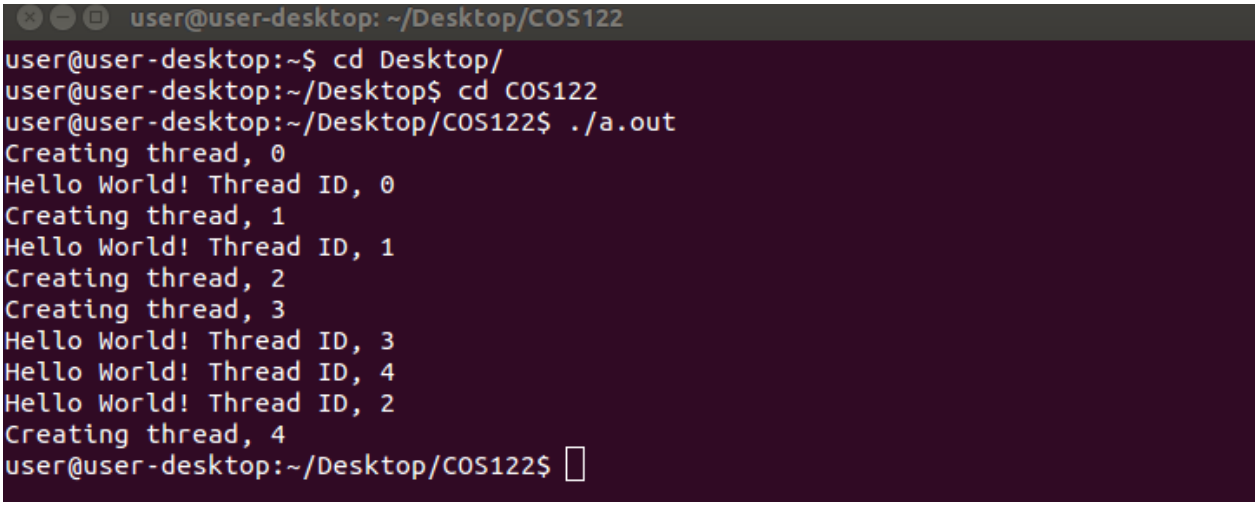
Task 3 - Threads [7 Marks]

Task 3 (a) - 2 marks

In this task, you are required to write simple C++ program. Your program should create 5 threads using *pthread_create*() function and display a message to show which thread was created. The newly created threads should also display a messages when they are created.
You can read about function pointers in C++ from here:

1. `http://www.newty.de/fpt/fpt.html`
2. `http://www.learncpp.com/cpp-tutorial/78-function-pointers/`
3. `http://www.cprogramming.com/tutorial/function-pointers.html`

Your output should look something like this.



Task 3 (b) - 5 marks

In this task, you are required to write a simple C++ program using a **struct** to pass multiple arguments to threads (namely, ID and data). Things to note:

- Your program should take the file name as a command line argument.
- For each thread created a message should be printed including the threads ID.
- Your program should create a thread for each line it reads from the file, and send that line to the thread to print as well as the threads ID.
- The thread should then print out the correct string. Remember this data has to be read and displayed in the order as seen in the file.
- You can create your own **.txt** file for your testing. On demo days, you will be given a new **.txt** for testing your code.

Your output should look something like this.

```
Creating Thread: 1149036945
Hello   [From Thread_ID: 1149036945]
Creating Thread: 1149036946
World,  [From Thread_ID: 1149036946]
Creating Thread: 1149036947
How     [From Thread_ID: 1149036947]
Creating Thread: 1149036948
are     [From Thread_ID: 1149036948]
Creating Thread: 1149036949
You     [From Thread_ID: 1149036949]
Creating Thread: 1149036950
Doing   [From Thread_ID: 1149036950]
Creating Thread: 1149036951
?       [From Thread_ID: 1149036951]
```

Task 4 - Threads, Static and Non-Static Member Functions [5]

In this task, you will be testing threads using static and non-static member functions. You are required to write a C++ program that passes a static member function to a *pthread_create*() function. Things to note in this task.

- You will be required to use function pointers in your code so I would advise you read about them if not done so in Task 3 already.
- Your program should create a new thread from a non static member function. Then use the *thread_ID* to create and execute the second thread for the static member function. This can help: `https://www.tutorialspoint.com/cplusplus/cpp_class_member_functions.htm`
- Your program should display an error in case the thread was not created.

Your output should look something like this

```
user@user-desktop:~/Desktop/COS122/static$ ./a.out
Executing thread 140334375220992 .
Task :: execute from Thread ID : 140334375220992
Waiting for thread 140334375220992 to execute.
Task :: threadFunc from Thread ID : 140334375220992
Exiting Main
user@user-desktop:~/Desktop/COS122/static$
```

Total: [22 Marks]