National University of Computer and Emerging Sciences, Lahore Campus

Course:	Operating Systems	Course Code:	CS205
Program:	BS(Computer Science)	Semester:	Fall 2019
Duration:	2 Hours	Total Marks:	45
Paper Date:	15-October-2019	Weight	
Section:	All Section	Page(s):	2
Exam:	Lab Mid Exam		

Instruction/Notes:
You must ensure that you have made proper submission of your code and well in time. No queries will be entertained later.
The code must be properly indented. You'll lose marks if it isn't.
Some questions have requirements. Ignoring any of the requirements will result in a ZERO.
Your submission file must bear your Roll number and the question number. Any submission not bearing a roll number will not be entertained even if it bears your name.
The submission path is /cactus/Xeon/Fall 2019/usman khan/OS/Mid submissions/Section X/Q1 or Q2; where X denotes your respective section.
Discussion with other students is not allowed.
Use of the mobile phones, old lab manuals, helping code, internet and flash drives is strictly prohibited.
You can use linux man pages though. The manuals for threads (pthread.doc), **files required for Q1 and Q2** have been provided in a folder located at /cactus/xeon/Fall 2019/usman khan/ OS
Plagiarism will result in F grade in lab.

Question 1 **(25 Marks)**

The "at" service in Linux allows a user to run a task(job) at some time in the future. The task-list can be given from the command line or a file. The tasks need to run only once in their lifetime and not periodically. For example, if we run the following command on terminal,

guest@optiplex$ echo "This text" | at 14:02

As a result, the command ' echo "This text" ' will be run the very next time the clock strikes 14:02(2:02 PM). The "at" command can even schedule tasks listed in a file.

You are required to write a very basic version of the "at" command, which will schedule the tasks specified in the file – tasks.txt. When "./myat" is run, it'll look for ***tasks.txt*** in the current directory. In case the file ***tasks.txt*** is not found, it should print "Unable to open tasks.txt!" to standard output (stdout) before exiting with an error. For every task found in ***tasks.txt***, a child will be forked. The child will be passed two things via an unnamed pipe, the time at which to run the task (hour in 24-hr format and minutes) and the name of the task to be scheduled. After all the tasks have been read, and all the children forked, and the respective time and commands sent to each and every child, "./myat" will exit.

On the other hand, each child will read the time and the task to execute from the pipe, and sit in loop waiting for the scheduled time to occur (although there are other sophisticated ways but we'll stick with a loop for this one time). As soon as the schedule time and the current time become equal, the command will run.

A function for comparing hours and minutes to the current time has been provided. The function, **short is_equal_to_current_time (unsigned short hour, unsigned short minutes)**, checks whether the provided value for the hour and minutes is equal to the current time or not.

Parameters

The function takes in two parameters, both of type unsigned short, hours and minutes. The value for the hours can range from 0 to 23 and the value for the minutes can range from 0 to 59.

Usage
is_equal_to_current_time (22, 12); //Checks whether the current time is equal to 22:12 (10:12 PM) or not.

Return Value
The function returns 1 if the given parameters are equal to the current time and zero if not. If the values for the parameters are not in the specified range, the function returns -1;

Sample tasks.txt	**Sample run and output of ./myat**(output on stdout after the scheduled time(s))
00 30 whoami 01 00 arch 18 30 ps	guest@optilplex$ ./myat guest@optiplex$guest x86_64 PID TTY TIME CMD 21665 pts/4 00:00:00 bash 21905 pts/4 00:03:02 ps

The format for ***tasks.txt*** is <hour in 24-hr format> <space> <minutes> <space> <taskname>. The first line of sample ***tasks.txt*** states that whoami command is to be scheduled at 00:30 (or 12:30 AM). The second line states that arch command is to be scheduled at 01:00 (01:00 AM). And the third line states that ps command is to be scheduled at 18:30 (6:30 PM).
For simplicity you can assume that the taskname has no further arguments.

You are required to submit a makefile along with your source code, that creates an output executable named - "myat".

You can assume that all times given in the file are correct (within specified range) and all commands mentioned are valid Linux commands.

Question 2 **(20 Marks)**

Write a program that will merge the content of different files in a single file. The names of the files will be received as a command-line argument. ***output.txt*** will contain content of all input files in order of the names of the files received via command-line.

oracle@ns2$ ./a.out File1.txt File2.txt

For example, running the above-mentioned command on the terminal will concatenates the contents of ***File1.txt*** and ***File2.txt*** in a single file ***output.txt.*** As ***File1.txt*** appears first in the command; hence, the contents of the ***File1.txt*** first in the ***output.txt***. Similarly, the contents of ***File2.txt*** appears after the content of ***File1.txt***.

File1.txt	***File2.txt***	***output.txt***
An operating system (OS) is system software that manages computer hardware, software resources, provides common services for computer programs.	A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by time-sharing, where the available processor time is divided between multiple processes.	An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by time-sharing, where the available processor time is divided between multiple processes.

The above command shows the working of the program for two files; however, you have to implement a program that can concatenates any number of files.

You have to create threads equal to the number of input files. Each thread will memory-map a single file whose name is received as an input parameter and return the pointer and size of memory-mapped region. After execution of all threads, the program will memory-mapped a file with the name ***output.txt***. If the file doesn't exist already the program should generate it. The program should copy all the contents from all the memory-mapped region of input files, in order of the names of the files received via command-line, into memory mapped region of ***output.txt.*** Make sure you unmapped all the memory regions you created earlier after the completion of the task.