Mini Tasks-1

Deadline : 17.08.2019 08:00 pm
Evaluation : 17.08.2019 09.30 pm

(Please avoid last minute submissions to avoid any hassles)

Languages Allowed: C/C++
Please post your queries on moodle if any.

Goal: To get familiar with read/write system calls (1) and get an idea about /proc in linux (2)

Note : system() or exec() family of functions are not allowed for any of the following below.

TASK 1

Write a program to copy files and measure the performance(time taken) to copy files in all the cases mentioned below.

1. Inside same partitions but different level(i.e. Hierarchy of folder structure) of folders.
2. Inside the same physical disk but different partitions.
3. From HDD to a USB drive or vice-versa.
4. Feature to split large files (> 4GB) and also to merge them when needed as an option.

You can only use Read and Write system calls for performing copy operation.

Test Cases for performance check

1. 1000 1KB files and 1 GB file to see the difference between copying multiple small files and a single large file.
2. A multimedia file.

Instructions :-

1. Find out the command to copy a file in Linux environment and understand its working.
2. Learn about system calls and how they work.

3. The input might be a very large text file or video file(5GB).Your code should be able to handle such large files.

Input :-

1. Calling format for the program :: ./myprog [src] [dest], names are excluding square brackets.
2. Detect split and merge requirements as per your logic.

TASK 2

Write a program which prints the following details for all processes or if provided a process id explicitly then for only that:

Note : Format for output must be clear and precise but can be custom as per your idea.

1. Filename of the executable
2. State – print R for running ,S for Sleeping ,D for sleeping in an uninterruptible wait, Z for zombie, T for traced or stopped
3. process id of parent process
4. session id
5. Environment variable details
6. File descriptors opened and associated with process.
7. Process root folder.

And also print the following system stats (not limited to) :

1. total number of context switches across all CPUs
2. number of processes currently running on CPUs.
3. number of processes currently blocked, waiting for I/O to complete.