OPERATING SYSTEMS 1

INSTRUCTOR: DR. SATHYA PERI
Hitesh Sehrawat - ma17btech11004@iith.ac.in

1.Design of the Program

This program determines the approx. time necessary to run a command from the command line.

I have used IPC mechanism and shared variables using the shared memory.The child process will write the starting time to a region of shared memory and after that execvp();. Then the child process will terminate,and the parent will read the starting time which is done using POSIX shared memory.

Used gettimeofday() to get time of the start_time and end_time of process to achieve the goal of the program which was to calculate the run time of command from the command line.

2.Working

In the program a child process is created using the fork() function which is defined in the header library “unistd.h”.Then it is used to execute the provided command.Just Before the child process is created and writes the start time in the shared memory.Parent process waits till the child process terminates done by “wait()”, and then end time is stored. Thus the difference between the start time and end time will give us the time elapsed for the provided command to run. Parent and child process communicate using the POSIX shared memory. “shm_open()” function creates shared memory for parent and child processes and the “ftruncate()” function allocated the memory of size of “struct timeval(16)”. “mmap” establishes the the memory-mapped file containing the shared memory object and returns a pointer which is used to access the shared memory.At the end the function “shm_unlink()” removes the shared memory space.

3.Analysis from various inputs

The table given below represents time elapsed for various Commands :

Commands	What command does?	Mean Elapsed Time of 10 consecutive trials (in seconds)
• **ls**	**Opens list of files in directory**	**0.003834s**
• **gnome-terminal**	**Opens terminal**	**0.145211s**
• **google-chrome**	**Opens google chrome**	**0.512666s ***
• **man fork**	**Opens man pages**	**~0.171536s ****
• **subl**	**Opens sublime**	**0.030199s**
• **rhythmbox**	**Opens rhythmbox**	**0.058650s***
• **mkdir foldername**	**Makes a folder of some folder name**	**0.003842s**
• **touch file**	**Makes a file of some filename**	**0.002143s**
• **python -V**	**Displays version of python**	**0.002023s**
• **firefox**	**Opens firefox**	**0.095159s***
• **clear**	**Clears the terminal screen**	**0.002228s**
• **ping -c 1 www.google.com**	**Used to test reachability of a host of an IP (1 time because of -c 1)**	**0.039292s**

***-** These apps were already opened in background so that to measure elapsed time. Else the time will appear after you close that tab manually that will show total elapsed for the time the tab was open.

****-** The elapsed time for “man” page is the time while it was open. So I tried to minimize this time elapsed by closing it as soon as a possible.

As clearly seen from the table the some commands which are used to run heavy applications take more time than other commands (and that too when they were in the cache).

I also used the command : $ time <command> in command line to calculate time of these above commands and then did some comparison.

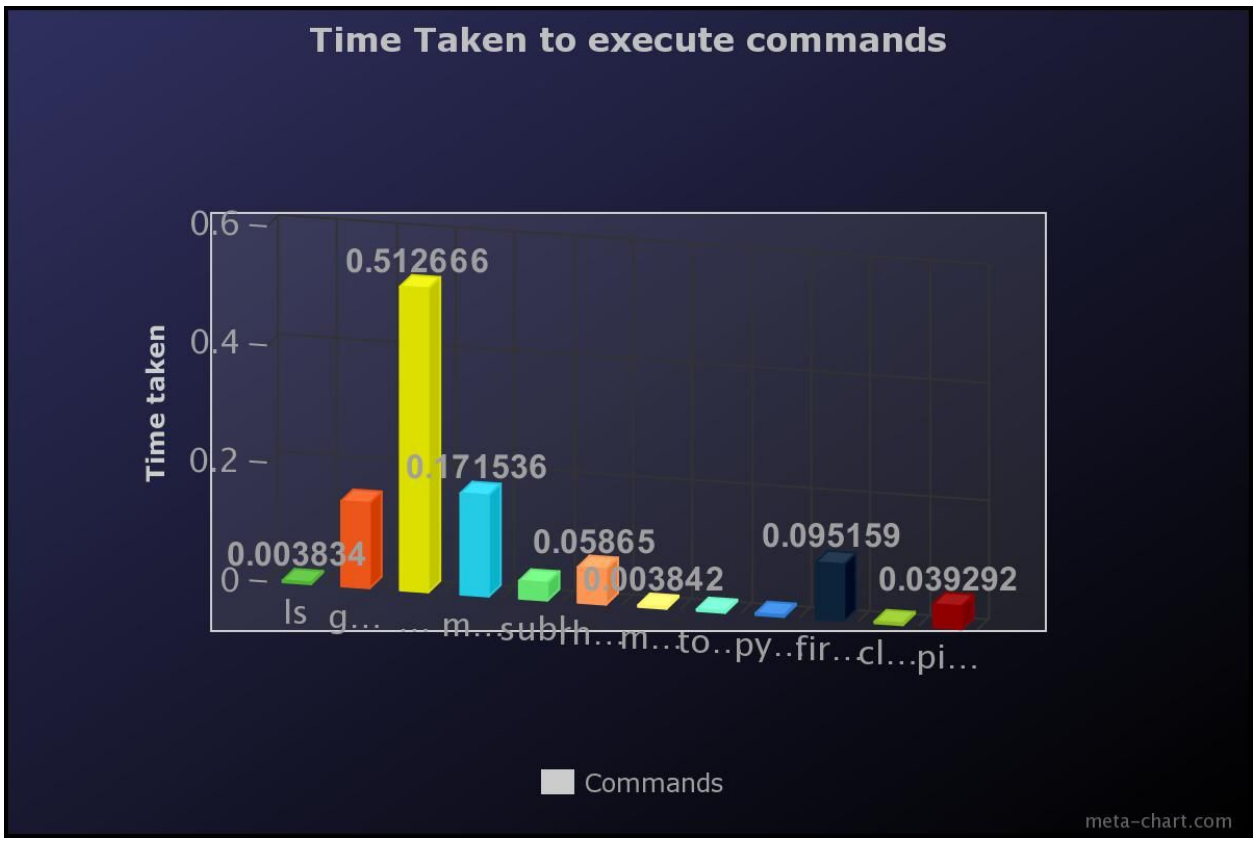
For eg : - $ time ls

This gives output as all files in directory followed by times given below :

real 0m0.005s //total time elapsed in executing the command

user 0m0.001s //time used by CPU in executing the process

sys 0m0.004s //time spent in system call within the kernel



Conclusion :

Time taken to execute a command essentially depends on the application which is run from the command line.Different processes take different time accordingly to the CPU and memory management.