

Operating Systems-1: CS3510

Autumn 2018

Programming Assignment 1 : Multi-Process Computation of Execution Time

Instructor : Dr. Sathya Peri

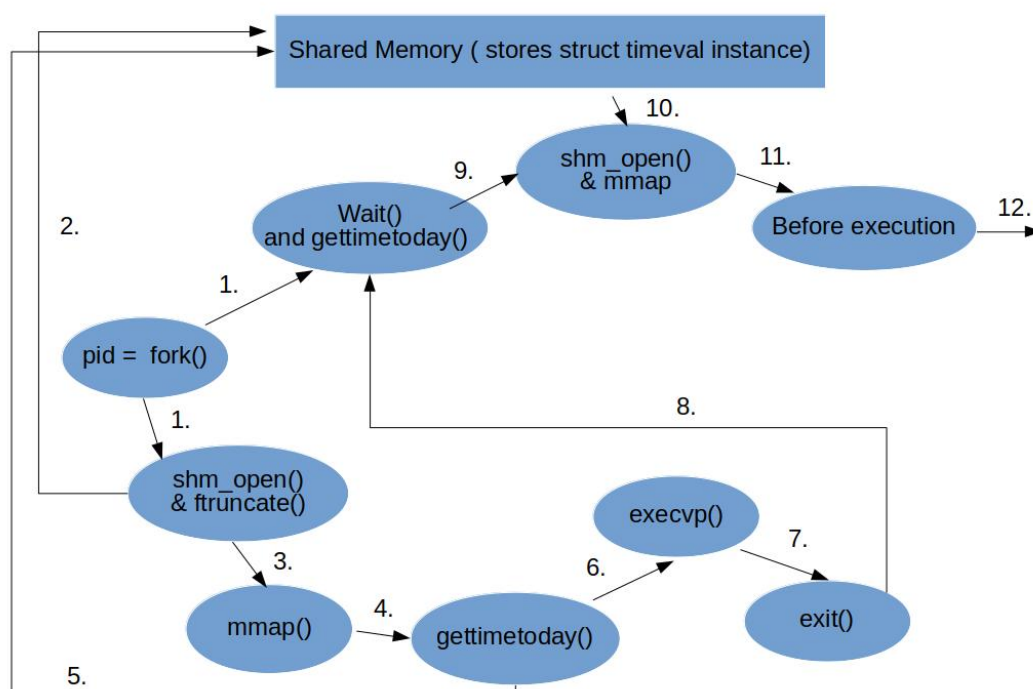
Author : Puneet Mangla (CS17BTECH11029)

1. Design of Program :

As soon as the child process is created , the child process creates a shared memory object. The size of the shared memory object is fixed and is equal to **sizeof (struct timeval)** because we only need to store before execution timestamp. After creating the shared memory the child process stores the before execution timestamp in shared memory for the parent process to access in future. Then it starts executing the command.

After child process terminates , parent process immediately records the timestamp and stores in other **struct timeval instance** . Then it extracts the before execution time from shared memory via **mmap** and calculates the elapsed time in seconds.

The visual design implementation is given below :



1. Creation of child process and parent starts waiting.
2. Child process creates shared memory instance.
3. Then it maps to shared memory which returns a pointer to shared memory instance of type **struct timeval*** .
- 4 and 5. Child process stores before execution timestamp using pointer it received .
- 6 and 7. Child process executes command and exits.
8. parent process starts execution and calculates **current** timestamp.
- 9 , 10 and 11. parent links with shared memory and gets before execution time.
12. Calculates the time elapsed.

2. Output Analysis :

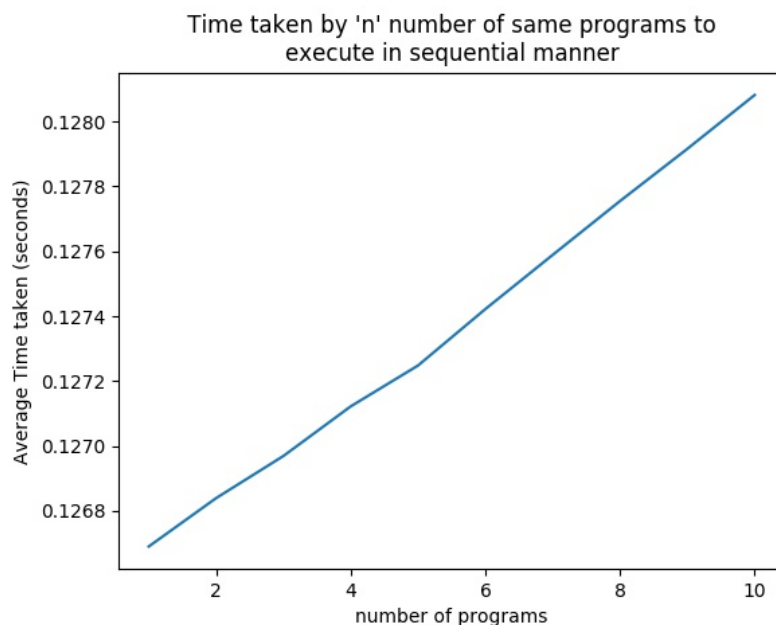
Since the elapsed time of command fluctuates every time , we took the average reading over **5 executions** of elapsed time for output analysis.

- **Average time for actual program**

Passing the same executable file to the arguments to same program 10 times , we obtained a graph showing execution time when 'n' same programs are executed in sequential manner. Graph obtained is linear conciding with our intutuion.

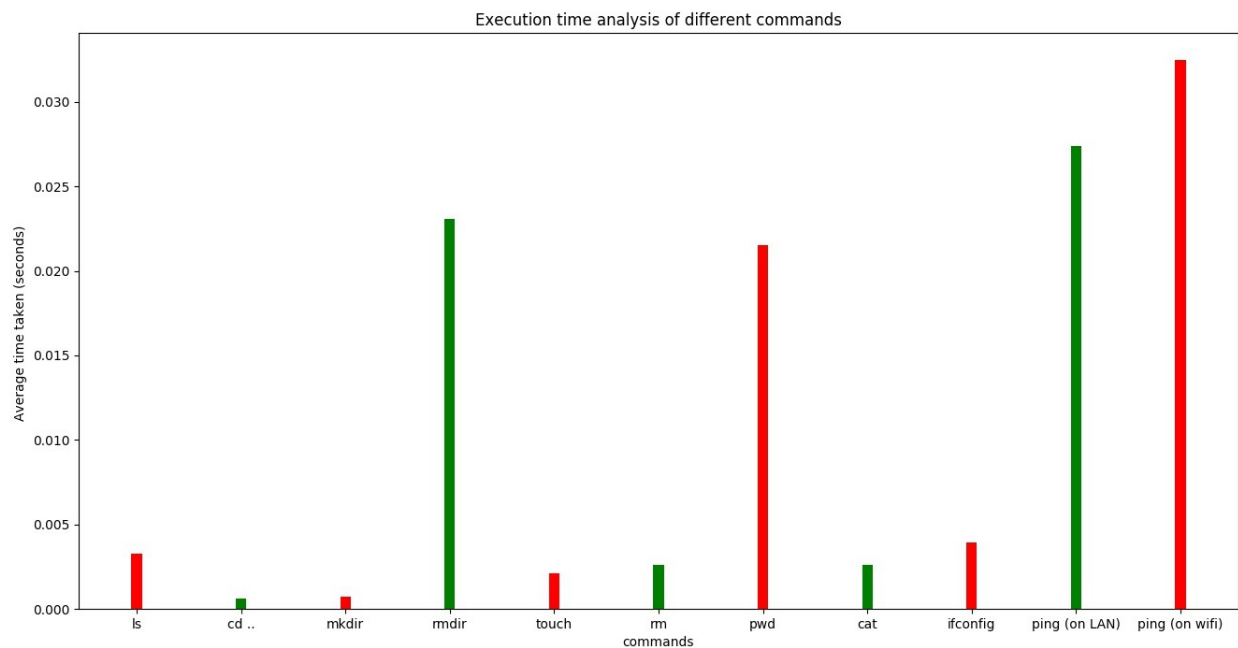
a.out a.out a.out 10 times

Note : need to remove shm_unlink to avoid segmentation fault.



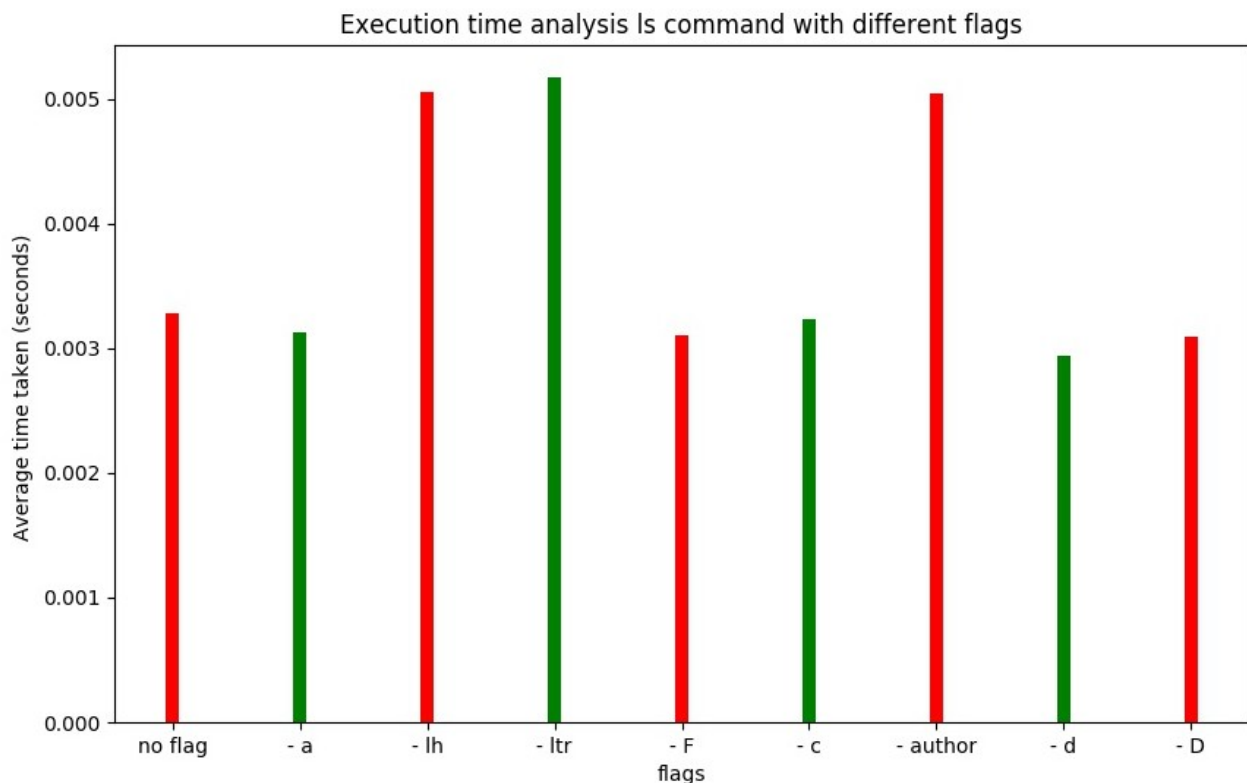
- **Average Execution time for standard linux commands.**

The average execution time of standard linux command are as follows



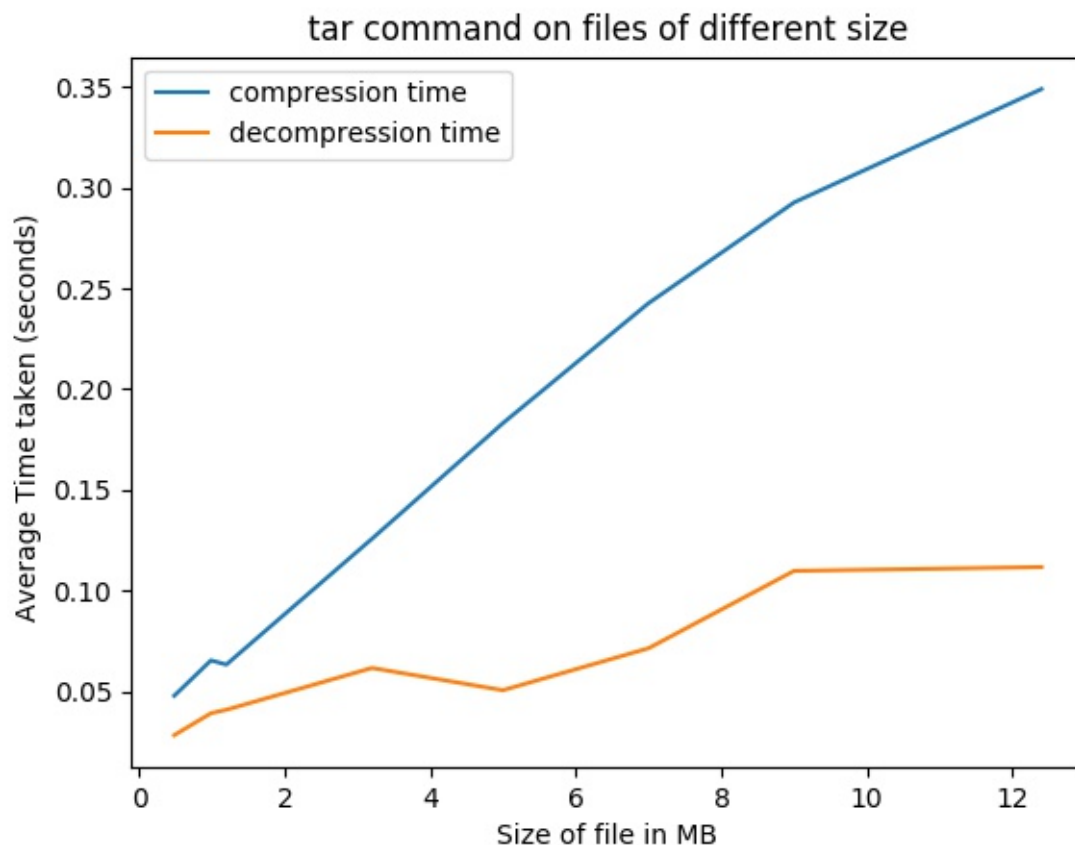
Since ping command depends on our internet speed , the time taken by it on wifi is larger than that of LAN.

- **Average Execution time of ‘ls’ command with different flags.**



From the graph we can see that flags like *lh,ltr,author* takes more time as compared to other flags. The reason is because the amount of information given by these flags more . For example flag *lh* gives **rx** permissions , date of creation , size , author etc. Which requires more execution.

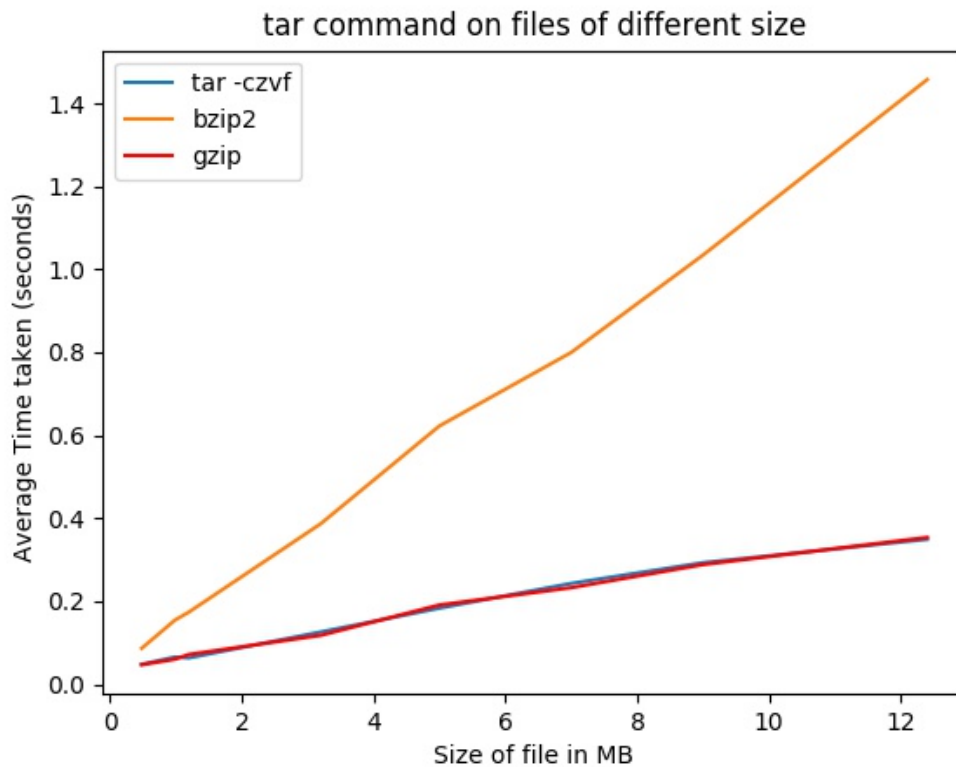
- **Analysis of ‘tar’ command on files of different sizes.**



Taking 8 books of different sizes and by using *tar* command , noted the archiving and decompression time. From the graph its clear that **compression time increases almost in linear fassion** as the size of file increases. **The time for decompression is much smaller than time of compression.**

- **tar -czvf , bzip2, gzip time comparison**

Compression time for various compression commands is given below.



The compression time of all the three commands follow a **linear fassion with size of file** . The interesting point is ***tar -czvf and gzip*** takes **same amount of time to compress files** which is much smaller than that of ***bzip2*** . So its good to use ***gzip and tar*** for file compression.