# Operating Systems–1: CS3510 Autumn 2018 Programming Assignment 1: Multi-Process Computation of Execution Time Report

Sai Harsha Kottapalli

November 15, 2018

## 1 Aim

Given a command (with its parameters if necessary), executes the command and outputs the time taken to execute the command.

## 2 Explanation of program

### 2.1 timeval

Here 'start' and 'end' are pointers which will store the values of start time of the command and end time of the command respectively.
The timeval structure is used to specify a time interval which has members - (tv_sec) i.e. time interval in seconds and (tv_usec) i.e. time interval in microseconds

### 2.2 mmap()

mmap() creates a new mapping in the virtual address space of the calling process.

If addr is NULL, then the kernel chooses the (page-aligned) address at which to create the mapping. For the next parameter we provide length of the mapping.

The prot argument describes the desired memory protection of the mapping (and must not conflict with the open mode of the file).
PROT_READ - Pages may be read.
PROT_WRITE Pages may be written.

The flags argument determines whether updates to the mapping are visible to other processes mapping the same region, and whether updates are carried through to the underlying file.
MAP_SHARED - Share this mapping. MAP_ANONYMOUS - The mapping is not backed by any file i.e. the fd argument is ignored.

As specified previously, fd argument is ignored. As some implementation require fd to be -1, we give that value.

Offset will be zero.

## 2.3 fork()

The pid_t data type represents process IDs. Each process has it's own process identifier which is a unique integer.

When we call fork() system call, a new process is created which has a copy of the address space of the original space. Here, the new process is the child process and the process which called fork() is referred to as parent process.

In the program, the child process has a copy of variables (with its data) namely - argc and argv, which are responsible for storing the number of arguments and the individual respectively.

The child process pid for the process becomes 0 while that of the parent becomes non-zero.

Later we use execvp() system call to replace the process's memory space with a new program.

## 2.4 execvp()

The first argument is a character string that contains the name of a file to be executed.
The second argument is a pointer to an array of character strings. More precisely, its type is char **, which is exactly identical to the argv array used in the main program.

Since argv[0] has the name of the binary file, we create the char **cmd which is responsible for storing the command and its parameters, so that we can passit it in the second argument easily.

It duplicates the actions of the shell in searching for an executable file.

This system call will load a binary file into memory (destroying the memory image of the program containing execvp() system call) and starts it's execution.

Since, the call to exec() overlays the process's address space with a new program, exec() does not return control unless an error occurs.

## 2.5 wait()

The parent process issues a wait() system call to move itself off the ready queue until termination of the child.

When the child process completes (by either implicitly or explicitly invoking exit()), the parent process resumes from the call to wait(), where it completes using the exit() system call.

## 2.6 gettimeofday()

We will record the current timestamp using this function. This function is passed a pointer to a struct timeval object, which contains two members: tv_sec and t_usec . These represent the number of elapsed seconds and microseconds since January 1, 1970

## 2.7 munmap()

The munmap() function shall remove any mappings for those entire pages containing any part of the address space of the process starting at addr(the

ones decalred through mmap()) and continuing for sizeof(the data type) bytes.

## 2.8 output

since we have called gettimeofday() just before execvp and just after wait() we approximately find the execution time of the given command , and show it in STDOUT.