CS3510 - Operating Systems – 1

Programming Assignment – 1: Computation of Execution time

MS17BTECH11013
P. Pavan Kalyan

A multi-process program involving IPS to find the time needed to execute an instruction from the command line.

Working Details:

Functions used:

fork()

prototype: `pid_t fork(void);`

fork() - returns a negative value means the **creation of a child process** was unsuccessful.

returns a zero to the newly created child process

returns a positive value, the process ID of the child process, to the parent. The returned process ID is of type pid_t.

shm_open

prototype: `int shm_open(const char*name, int o_flag, mode_t mode);`

it creates and opens a new/existing POSIX shared memory object. It takes in argument the name of the memory object you want to create.O_flag is a bit mask which will tell the type of permission it should has

Ftruncate

Prototype: `int ftruncate(int fd, off_t length)`

It truncates the size of the shared memory (file des. as argument) to the desired size as passed as an argument (length as argument)

On success zero, is returned. On error -1 is returned.

Mmap

Prototype: `void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);`

It creates a new mapping in the virtual address space of the calling process. Generally we pass 0/NULL as the first argument. Length in this case is the length of the struct timeval because we want to share the data from child process which is of type struct timeval. Prot means protection for the shared memory of the mapping. The arguments in this case are PROT_READ, PROT_WRITE means read and write acess in a protected manner. Flags tell whether updates to the mapping should be visible to other process. MAP_SHARED gives the sharing of mapping as we want to share it with the memory. Fd argument value is kept as -1 in general for most programs. Offset value is 0.

gettimeofday

`prototype: int gettimeofday( struct timeval *tv, struct timezone *tz);`

It will save the current time in the struct timeval tv. This function requires a pointer to struct timeval tv as an argument. Struct timeval has two arguments tv_sec and t_usec which gives the time elapsed in seconds and micro-seconds since January 1, 1970.

execvp

prototype: `int execvp(const char*file, char *const argv[]);`

execvp provide an array of pointer to null-terminated strings that represent the argument list available to the new program. 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. We know that exec replaces the process address space with new program, exec does not return control unless an error occurs.

Wait

Prototype: `pid_t wait(int *wstatus);`

This command makes the parent process wait until the child process terminates. Given in the assignment that we have to first execute the child process first so in the parent process we call wait(). The parent resumes from the call to wait().

munmap

prototype: `int munmap(void *addr, size_t length);`

it deletes the mappings for the specified address range and causes further referencs to address within the range to generate invalid memory references.

shm_unlink

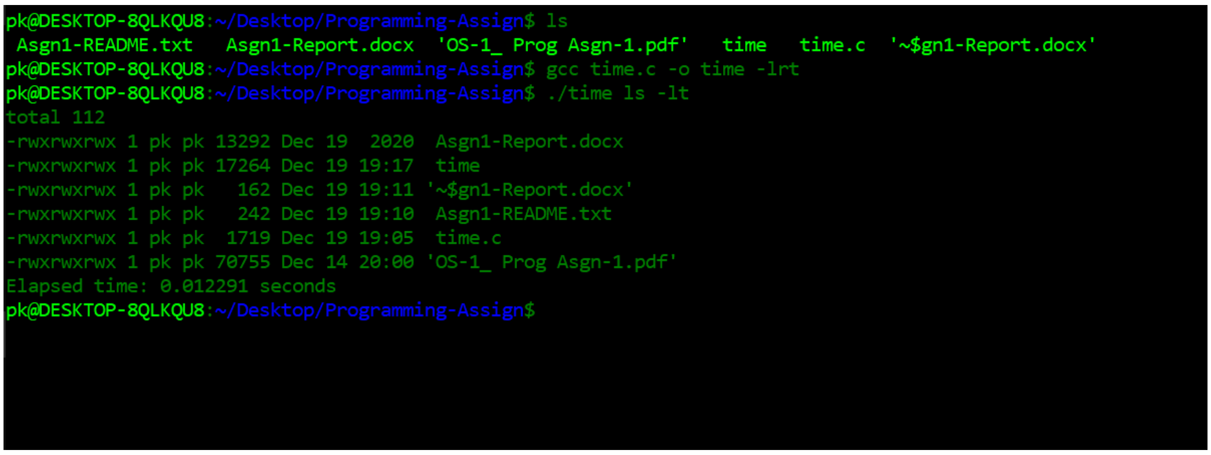
prototype: `int shm_unlink(const char *name);`

logic used:

There are two variables start and end of type struct timeval. As our aim is to record the time taken to execute the command. We need to know the time before the start of the command and after the end of the command. This can be achieved by sharing the start time which is the time before the execution of the command in the child process to the parent using the shared memory. After execution of the command we enter into parent process we then record the time which we call it end which signifies the end of execution of command. We retrive the time data from the shared memory and subtract it with end. As the struct timeval gives both elapsed second and microseconds we need to subtract from start from end. The formula is

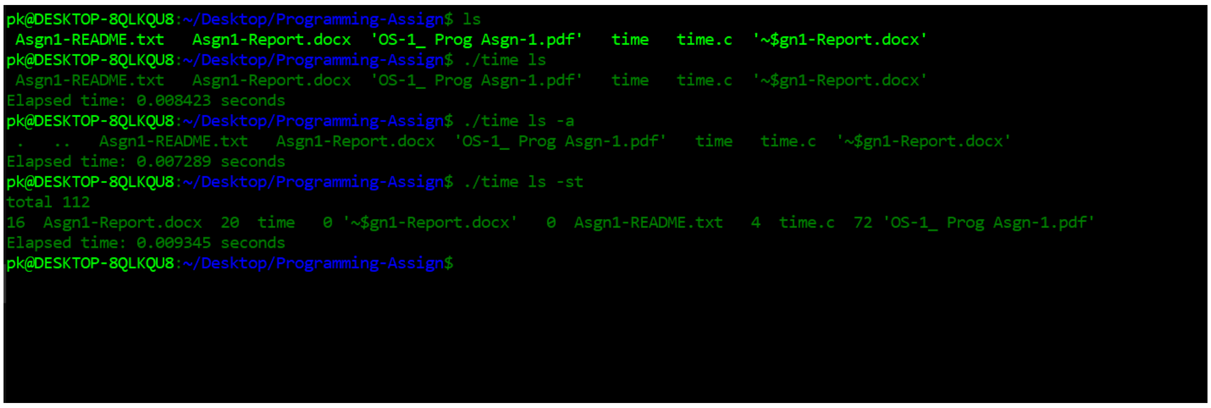
Time taken = (end.tv_sec – start.tv_sec) + (end.tv_usec – start.tv_usec)/1000000;

output:



```
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ls
 Asgn1-README.txt   Asgn1-Report.docx  'OS-1_ Prog Asgn-1.pdf'   time   time.c  '~$gn1-Report.docx'
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ gcc time.c -o time -lrt
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ./time ls -lt
total 112
-rwxrwxrwx 1 pk pk 13292 Dec 19  2020  Asgn1-Report.docx
-rwxrwxrwx 1 pk pk 17264 Dec 19 19:17  time
-rwxrwxrwx 1 pk pk   162 Dec 19 19:11 '~$gn1-Report.docx'
-rwxrwxrwx 1 pk pk   242 Dec 19 19:10  Asgn1-README.txt
-rwxrwxrwx 1 pk pk  1719 Dec 19 19:05  time.c
-rwxrwxrwx 1 pk pk 70755 Dec 14 20:00 'OS-1_ Prog Asgn-1.pdf'
Elapsed time: 0.012291 seconds
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$
```

Trying for various other commands and checking the output elapsed time



```
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ls
 Asgn1-README.txt   Asgn1-Report.docx  'OS-1_ Prog Asgn-1.pdf'   time   time.c  '~$gn1-Report.docx'
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ./time ls
 Asgn1-README.txt   Asgn1-Report.docx  'OS-1_ Prog Asgn-1.pdf'   time   time.c  '~$gn1-Report.docx'
Elapsed time: 0.008423 seconds
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ./time ls -a
 .   ..   Asgn1-README.txt   Asgn1-Report.docx  'OS-1_ Prog Asgn-1.pdf'   time   time.c  '~$gn1-Report.docx'
Elapsed time: 0.007289 seconds
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$ ./time ls -st
total 112
16  Asgn1-Report.docx  20  time   0 '~$gn1-Report.docx'   0  Asgn1-README.txt   4  time.c  72 'OS-1_ Prog Asgn-1.pdf'
Elapsed time: 0.009345 seconds
pk@DESKTOP-8QLKQU8:~/Desktop/Programming-Assign$
```