Vinta Reethu
Operating Systems
Indian Institute of Technology Hyderabad
Assignment-1

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Programming Assignment-1

Vinta Reethu - ES18BTECH11028

December 19, 2020

Question :

Multi-process program involving IPC to find the time needed to execute an instruction from the command line.

Working details :

Header files :

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <unistd.h>
```

These are the header files that are added in the written c file. These are necessary for us to perform the operations in this code.

Main function :

```
int main(int argc, char** argv)
```

- **main()** has two arguments. It takes the inline arguments. The first argument is the number of command line arguments and second is char array of the command line arguments.

```
if(argv[1]==NULL)
{
    printf("Error! Please provide an argument. \n");
    return 0;
}
```

- If no arguments are passed then we exit the program with an error message.

Vinta Reethu
Operating Systems
Indian Institute of Technology Hyderabad
Assignment-1

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

```c
const int SIZE = 512;                    // Specifying size of the shared memory object
const char *name = "OS_Assign1";         // Name of the shared memory object
int fd;                                  // File descriptor for shared memory object
char *ptr;                               // Pointer to the shared memory object
```

- The parent and child are separate process, so in order to share the time value between them we need to provide a **shared memory** object. The above code has the variable declarations for creating this shared memory object.

```c
pid_t pid;                               // Creating a pid variable
```

- Next we create a process identifier variable (pid). This variable is for process identification.

```c
struct timeval start_time,end_time;   // Stores time when declared
```

- **timeval** is a structure that has two arguments namely tv_sec,tv_usec. This variable is declared to store the time in it.

```c
fd = shm_open(name, O_CREAT | O_RDWR, 0666);   // Create a shared memory object
```

- **shm_open** creates and opens a new, or opens an existing, shared memory object. A POSIX shared memory object is used to a handle which can be used by unrelated processes to mmap the same region of shared memory.

```c
ftruncate(fd, SIZE);                 // Configuring the size of the shared memory object
```

- **ftruncate()** is used to truncate a file to a specified length. Here the size is given to be 512.

```c
ptr = (char *)mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
// Mapping the memory of shared memory object
```

- **mmap()** creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in addr. The length argument specifies the length of the mapping (which must be greater than 0)

```c
pid = fork();                            // Fork a child process
```

- **fork()** creates a new process by duplicating the calling process. The new process is referred to as the child process. The calling process is referred to as the parent process.
- Once fork is called you will have 2 processes running. The child process gets the pid 0 and parent gets a pid greater than 0.

```c
if (pid < 0)                             // Error occurred, if pid is less than zero
{
    fprintf(stderr, "Fork Failed");
    return 1;
}
```

- If the value in pid is less than zero, some error has occurred hence we report it and quit the program.

Vinta Reethu
Operating Systems
Indian Institute of Technology Hyderabad
Assignment-1

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Child Process :

```
gettimeofday(&start_time, NULL); // Save the current time
sprintf(ptr,"%ld %ld",start_time.tv_sec,start_time.tv_usec); // Writing the current
    time to the created memory object
if(execvp(argv[1], argv+1) < 0)  // If execvp outputs 0, it couldn't execute the
   given command; else it executes it
{
    printf("Could not execute the given command! \n");
}
exit(0);
```

- First we save the current time in the timevalue structure. The stored time is elapsed from Jan 1st 1970. We use **sprintf** to write the values because the shared memory object is pointed by a char pointer.
- We write this time present in the timevalue structure to the saved memory object.
- Next we execute the command given from the command line using **execvp**.
- While executing the **execvp** command if we obtain any error we print it and exit the child process.

Parent Process :

```
wait(NULL);                          // Wait till child process terminates
```

- With **wait()** command we make parent process wait till the child terminates.

```
gettimeofday(&end_time, NULL);    // Save the current time
long int start_sec,start_microsec;
sscanf(ptr,"%ld %ld",&start_sec,&start_microsec); // Scanning the saved variables
   from memory object
double elapsed = (end_time.tv_sec-start_sec + (end_time.tv_usec-start_microsec)
   *0.000001); // Finding out the difference between start and end time
printf("Elapsed time: %lf\n",elapsed); //Printing the elapsed time to user
```

- Again we save the current time in a timevalue structure.
- By using **sscanf** we read the values from the shared memory object.
- Next we calculate the elapsed time by subtracting the current time with the previously recorded time and print it.

```
shm_unlink(name);
```

- **shm_unlink()** function removes an object previously created by shm_open().

Vinta Reethu
Operating Systems
Indian Institute of Technology Hyderabad
Assignment-1

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Output Analysis :

- The elapsed time is in the order's of **0.01 secs**.
- If we increase the size of memory objects the elapsed time for the same instruction is increasing.
- Special builtin functions like **"cd"** can't be executed as they are not executable, hence execvp fails.