COMPSCI 3SH3 LAB 2 REPORT

Shawn Li, lix229

February 9, 2020

Contents

1 Question 1: Shared memory approach

1.1 Module Description

The module time writes the up time of the program to a shared memory it creates, then reads from it, and display the result on the screen. Time is measured with the help of `sys/time.h`.

1.2 Source code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/time.h>
int main(int argc, char **argv )
{
```

```
typedef struct timeval timeval_t;
/* the size (in bytes) of shared memory object */
const int SIZE = 4096;
/* name of the shared memory object */
const char *name = "time";
/* shared memory file descriptor */
int fd;
/* pointer to shared memory obect */
timeval_t *ptr;
/* create the shared memory object */
fd = shm_open(name,O_CREAT | O_RDWR, 0666);
/* configure the size of the shared memory object */
ftruncate(fd, SIZE);
/* memory map the shared memory object */
ptr = (timeval_t *)
mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
pid_t pid;
pid = fork();
    if (pid < 0) {
    /* error occurred */
    fprintf(stderr, "Fork Failed"); return 1;
}
else if (pid == 0) {
    gettimeofday(ptr,0);
    execvp(argv[1], argv + 1);
}
else {
    wait(NULL);
    timeval_t endtime;
    gettimeofday(&endtime,0);
    timeval_t starttime;
    starttime = *ptr;
    shm_unlink(name);
    timeval_t elapsed_time;
    timersub(&endtime, &starttime, &elapsed_time);
    printf( "\nElapsed time: %lu.%06lu seconds\n", elapsed_time.tv_sec, elapsed_ti
    printf( "Command: " );
    char ** arg = argv + 1;
    while( *arg ) printf("%s ", *arg++);
    printf("\n");
```

```
    }
    return 0;
}
```

2 Question 2: Pipe approach

2.1 Module Description

The module pipe does the same thing except using a pipe instead.

2.2 Source code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/time.h>
#define READ_END 0
#define WRITE_END 1
int main(int argc, char **argv )
{
    typedef struct timeval timeval_t;
    int pipeFD[2];
    pipe(pipeFD);
    pid_t pid;
    pid = fork();
        if (pid < 0) {
        /* error occurred */
        fprintf(stderr, "Fork Failed"); return 1;
    }
    else if (pid == 0) {
        close(pipeFD[READ_END]);
        timeval_t startTime;
        gettimeofday(&startTime, 0);
        write(pipeFD[WRITE_END], &startTime, sizeof(startTime));
        close(pipeFD[WRITE_END]);
        execvp(argv[1], argv + 1);
```

```
    }
    else {
        wait(NULL);
        timeval_t end_time;
        gettimeofday( &end_time,0);
        timeval_t startTime;
        close(pipeFD[WRITE_END]);
        read(pipeFD[READ_END], &startTime, sizeof(startTime));
        close(pipeFD[READ_END]);
        timeval_t elapsed_time;
        timersub( &end_time, &startTime, &elapsed_time );
        printf( "\nElapsed time: %lu.%06lu seconds\n", elapsed_time.tv_sec, elapsed_tir
        printf( "Command: " );
        char ** arg = argv + 1;
        while( *arg ) printf("%s ", *arg++);
        printf("\n");
    }
    return 0;
}
```