MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY, ALLAHABAD

Computer Science and Engineering Department

Name – Kamani Kumari

Class – MCA 1st semester

Reg No. – 2024CA048

Subject – Shell Programming Lab

Submitted To – Dr. Ranvijay Sir

Assignment-5

Date.....

Exercise 20

Ques :- Write a c program that illustrates the creation of a child process using fork System call.

```
#include <stdio.h>
#include <stdlib.h>
#include<unistd.h>

int main() {
    Pid_t pid;
    pid = fork();
if (pid < 0) {
    fprint(stderr, "fork failed.\n");
    return 1;
}
else if (pid == 0) {
    printf("This is the child process. PID = %d\n", getpid());
}

else {
  printf("This is the parent process. PID = %d, child PID =
    %d\n", getpid(), pid);
}

  return 0;
}
```

Ganpati

Date.....

Exercise 21

Write a C program that illustrates for file locking using semaphores.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <string.h>

#define FILE_NAME "Shared_file.txt"
#define SEM_NAME "/file_semaphore"

void write_to_file( const char *message ){
    FILE *file = fopen( FILE_NAME, "a");
 if( file == NULL){
        perror( "Failed to open file");
        return;0
 }
 fprintf( file, "%s\n", message);
  fclose( file);
 }

int main(){
        sem_t *Semaphore;
 Semaphore = sem_open( SEM_NAME, O_CREAT, 0644, 1);
 if( Semaphore == SEM_FAILED){
```

Ganpati

Date.....

```
    perror ("failed to open Semaphore");
    exit (EXIT_FAILURE);
}
printf ("Trying to acquire lock...\n");
if (sem_wait (semaphore) < 0) {
    perror ("Failed to lock Semaphore");
    sem_close (semaphore);
    exit (EXIT_FAILURE);
}
printf ("lock acquired. writing to file...\n");
write_to_file ("Process is writing to the file.");
printf ("File write Complete. Relasing lock...\n");
if (sem_post (semaphore) < 0) {
    perror ("Failed to unlock Semaphore");
    sem_close (semaphore);
    exit (EXIT_FAILURE); }
if (sem_close (semaphore < 0) {
    perror ("Failed to close semaphore");
    exit (EXIT_FAILURE);
}
    sem_unlink (SEM_NAME);

printf ("lock released and program complete.\n");

    return 0;
}
```

Ganpati

Date.....

Exercise 22

Write a C program that implements a producer Consumer System with two processes (using semaphores).

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <sys/wait.h>

#define BUFFER_SIZE 5
#define PRODUCE_COUNT 10
typedef struct {
        int buffer[ BUFFER_SIZE];
        int in;
        int out;
} shared_data_t;
int main() {
        int i;
        pid_t pid;
  shared_data_t *shared_data = mmap(NULL, sizeof(shared_data_t),
   PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS,
       -1, 0);
if( shared_data == MAP_FAILED) {
        perror ("mmap failed");
        exit ( EXIT_FAILURE); }
```

Ganpati

__/__/__ Date.....

```
        shared_data -> in = 0;
        shared_data -> out = 0;
Sem_t *sem_empty = sem_open("/sem_empty", O_CREAT | O_EXCL,
                            0644, BUFFER_SIZE);
Sem_t *sem_full = sem_open("/sem_full", O_CREAT | O_EXCL, 0644, 0);
Sem_t *sem_mutex = sem_open("/sem_mutex", O_CREAT | O_EXCL, 0644, 1);

if( sem_empty == SEM_FAILED || sem_full == SEM_FAILED ||
    sem_mutex == SEM_FAILED ){
        perror("Semaphore initialization failed");
    exit( EXIT_FAILURE);
}
    pid = fork();
if( pid <0){
        perror("Fork failed");
        exit( EXIT_FAILURE);
    }
if( pid ==0){
    if( i=0; i< PRODUCE_COUNT; i++){
            sem_wait( sem_full);
            sem_wait( sem_mutex);
    int item = shared_data -> buffer[shared_data -> out];
    printf("Consumer consumed: %d\n", item);
shared_data -> out = (shared_data -> out +1) % BUFFER_SIZE;
sem_post( sem_mutex);
sem_post( sem_empty);
    sleep(1); } }
```

Ganpati

Date.....

```
else{
    for (i=0; i< PRODUCE_COUNT; i++){
        int item = i+1;
    sem_wait(sem_empty);
    sem_wait (sem_mutex);
Shared_data -> buffer[shared_data -> in] = item;
printf ("Producer Produced : %d \n", item);
    shared_data -> in = (shared_data -> in+1) % BUFFER_SIZE;

sem_post (sem_mutex);
sem_post (sem_full);
    sleep(1);
}
    wait(NULL);
    sem_close (sem_empty);
    sem_close (sem_full);
    sem_close (sem_mutex);
    sem_unlink ("/sem_empty");
    sem_unlink ("/sem_full");
    sem_unlink ("/sem_mutex);

munmap( shared_data,  sizeof(shared_data_t));
}
    return 0;

}
```

Ganpati

Date

Exercise 23

Write a C program that illustrates inter process communication using shared memory system calls.

```
#include< stdio.h>
#include <stdlib.h>
#include<string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/type.h>
#include< unistd.h>
        #define SHM_SIZE 1024
int main(){
        key_t key = frok("shmfile",65);
        int shmid;
  if ((shmid = shmget(key, SHM_SIZE, 0666|IPC_CREAT))<0){
        perror("shmget failed");
        exit(EXIT_FAILURE);
    }
  pid_t pid = fork();
 if( pid<0){
        perror("Fork failed");
        exit(EXIT_FAILURE);
    }
 if( pid==0){
        sleep(1);
    char *date = (char *) shmat(shmid, void *)0, 0);
        perror("shmat failed");
```

Ganpati

```
    exit( EXIT_FAILURE);  }
else {
    char *data =
     Const char *message = "Hello from the writer process!";
printf( "writer: writing to shared memory: \"%s\"\n", message);
 Strncpy( data, message, SHM_SIZE);

if ( shmdt(data) == -1){
        perror ("shmdt failed");
         exit( EXIT_FAILURE); }
wait(NULL);
 if ( shmctl( shmid, IPC_RMID, NULL) == -1){
      perror ("shmctl failed");
       exit ( EXIT_FAILURE); }
   }
  }

   return 0;
}
```

Exercise 24

Write a program in C that illustrates how to execute two Command Concurrently with a Command pipe.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int pipefd[2];
    pid_t pid1, pid2;

if( pipe(pipefd == -1) {
    perror ("Pipe failed");
    exit (EXIT_FAILURE);
}
    pid1 = fork();
if( pid1 <0) {
    perror ("fork failed");
    exit (EXIT_FAILURE);
}
if( pid1 == 0) {
    close (pipefd[0]);
    dup2 (pipefd[1], STDOUT_FILENO);
    close (pipefd[1]);

execlp ("execlp failed");
    exit (EXIT_FAILURE);
}
```

```
    pid2 = fork();
if( pid2 < 0){
      perror("fork failed");
       exit( EXIT_FAILURE);
  }
if( pid2 == 0){
       close( pipefd[1]);
    dup2( pipefd[0], STDIN_FILENO);
      close( pipefd[0]);

   execlp("wc", "wc", "-l", NULL);
    perror("Execlp failed");
      exit( EXIT_FAILURE);
   }
      close( pipefd[0]);
      close( pipefd[1]);

 waitpid( pid1, NULL, 0);
 waitpid( pid2, NULL, 0);

 return 0;

}
```