Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

17 February 2020

UCS1411 - Operating Systems Laboratory

Exercise – 5-InterProcess Communications using Shared Memory

Objective:

Develop the following applications that uses interprocess communication concepts using shared memory.

Code:

Q1.Develop an application for getting a name in parent and convert it into uppercase in child using shared memory.

```
#include <sys/ipc.h>
#define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
```

```c
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>
#include<ctype.h>


int main(){
    int pid;

    //Input buffer in parent
    char *parent_ip;
    parent_ip=(char*)malloc(100*sizeof(char));

    //Output buffer in child
    char *child_op;
    child_op=(char*)malloc(100*sizeof(char));

    //Creating shared memory location
    int shmemid=shmget(146,100,IPC_CREAT|00666);

    pid=fork();

    //Parent process
    if(pid>0){
        printf("\nParent Process:\n");

        //Attaching memory location
        parent_ip=shmat(shmemid,NULL,0);

        printf("\nEnter a string: \n");scanf(" %[^\n]",
   parent_ip);
        wait(NULL);
        shmdt(parent_ip);

    }
    //Child Process
    else{
        sleep(5);

        printf("\nChild Process:\n");

        //Attaching memory location
        child_op=shmat(shmemid,NULL,0);

        printf("\nEntered string: %s\n",child_op);

        printf("\nString in upper case: ");
        int i=0;
        while(child_op[i]){
            putchar(toupper(child_op[i]));
            i++;
        }
        printf("\n");
        shmdt(child_op);
    }
    shmctl(shmemid,IPC_RMID,NULL);
}
```

Output:

```
Parent Process:

Enter a string:
hello

Child Process:

Entered string: hello

String in upper case: HELLO
```

Q2.Develop an client / server application for file transfer using shared memory.

```
#include <sys/ipc.h>
#define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>
#include<fcntl.h>
```

```c
#include<ctype.h>

int main(){

    printf("\nClient: \n");
    //Creating shared memory location
    int shmemid=shmget(146,1000,IPC_CREAT|00666);

    //Buffer to read from/write onto the shared memory location
    char *file_read=(char*)calloc(1000,sizeof(char));

    //Attaching to shared memory
    file_read=shmat(shmemid,NULL,0);

    printf("\nEnter file name: ");
    scanf(" %s",file_read);

    //Sleeping to get contents of file from server
    sleep(5);
    printf("\nFile Contents:\n");
    printf("\n%s\n",file_read);

    //Detaching from shared memory
    shmdt(file_read);

    //Deallocating the acquired memory location
    shmctl(shmemid,IPC_RMID,NULL);
}
```

```c

#include <sys/ipc.h>
#define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>
#include<fcntl.h>
#include<ctype.h>

int main(){

    printf("\nServer: \n");
    //Creating shared memory location
    int shmemid=shmget(146,1000,IPC_CREAT|00666);

    //Buffer to read from/write on the shared memory
    char *file_read=(char*)calloc(1000,sizeof(char));

    //Sleeping to accept file name as input from client
    sleep(2);

    //Attaching to shared memory
    file_read=shmat(shmemid,NULL,0);
    printf("\nFile to be read: %s\n",file_read);

    //Opening file to be read
    int sourcefd=open(file_read,O_RDONLY);
    if(sourcefd==-1){
        printf("No source file");
    }
    else{
        //Reading file contents onto the buffer
        int readfd=read(sourcefd,file_read,100);
        file_read[readfd]=’\0’;
        printf("\nFile read successfully\n");
        wait(NULL);
        close(sourcefd);
        //Detaching buffer from shared memory
        shmdt(file_read);
    }

}
```

Output:

```
Client:

Enter file name: source.txt

File Contents:

asdfgf ;lkjhj
shiva
sharvan
shashu
siddharth
```

```
Server:

File to be read: source.txt

File read successfully
```

Q3. Develop an client/server chat application using shared memory.

```c

#include <sys/ipc.h>
#define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>
#include<fcntl.h>
#include<ctype.h>

int main(){
    //Creating shared memory location
    int shmemid=shmget(146,1000,IPC_CREAT|00666);

    //Buffer to read from/write onto the shared memory location
    char *file_read=(char*)calloc(1000,sizeof(char));

    //Attaching to shared memory
    file_read=shmat(shmemid,NULL,0);

    while(strcmp(file_read,"end")!=0){

        //Sleeping to get contents of file from server
        printf("\nServer:%s\n",file_read);
        printf("\nYou: ");scanf(" %[^\n]",file_read);
        sleep(5);
        wait(NULL);
    }
    //Detaching from shared memory
    shmdt(file_read);

    //Deallocating the acquired memory location
    shmctl(shmemid,IPC_RMID,NULL);
}
```

```c

#include <sys/ipc.h>
#define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>
#include<fcntl.h>
#include<ctype.h>

int main(){
    //Creating shared memory location
    int shmemid=shmget(146,1000,IPC_CREAT|00666);

    //Buffer to read from/write onto the shared memory location
    char *file_read=(char*)calloc(1000,sizeof(char));

    //Attaching to shared memory
    file_read=shmat(shmemid,NULL,0);

    while(strcmp(file_read,"end")!=0){

        //Sleeping to get contents of file from server
        printf("\nClient:%s\n",file_read);
        printf("\nYou: ");scanf(" %[^\n]",file_read);
        sleep(5);
        wait(NULL);

    }
    //Detaching from shared memory
    shmdt(file_read);

    //Deallocating the acquired memory location
    shmctl(shmemid,IPC_RMID,NULL);
}
```

Output:

```
Client:

You: hi client

Client:hi server

You: hello client

Client:how r u

You: how u doinhh

Client:ggg

You: how about u

Client:im fine

You: nice
```

```

Server:hi client

You: hi server

Server:hello client

You: how r u

Server:hhow u doingg

You: ggg

Server:how about u

You: im fine

Server:nice

You: end
```