Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 5 – Interprocess communication

Date: 11/04/2022

Aim:

To develop the following applications that use interprocess communication concepts using shared memory:

1. An application for getting a name in parent and converting it into uppercase in child.
2. A client/server application for file transfer.
3. A client/server chat application.

Algorithm:

1) Application for converting to uppercase in child:

1. Start
2. Get the shared memory identifier.
3. Get the process id by forking.
4. If the process id is positive, then do the following:
    1. Attach a variable name to the shared memory.
    2. Get the name as input from the user.
    3. Detach the variable from the shared memory.
5. If the process id is 0, then do the following:
    1. Attach a variable to the shared memory.
    2. Convert the given name to uppercase.
    3. Print the name in uppercase.
    4. Detach the variable from the shared memory.
6. Remove the shared identifier and destroy the segment.
7. Stop

2) Application for file transfer:

Client side:

1. Start
2. Get the key for the shared memory by forking.
3. Get the shared memory identifier.
4. Attach a variable to the shared memory.
5. Get file name from the user.
6. Print the contents of the file onto the server.
7. Stop.

Name: Krithika Swaminathan
Roll No.: 205001057

Server side:

1. Start
2. Get the key for the shared memory by forking.
3. Get the shared memory identifier.
4. Read the file name from the shared memory.
5. Read the contents of the file using a pointer.
6. Print the contents of the file.
7. Close the file.
8. Detach from the shared memory.
9. Destroy the shared memory space.
10. Stop

3) Chat application:

Client side:

1. Start
2. Get the process id and generate a shared memory identifier.
3. Set the key to some value.
4. Attach the structure to the shared memory.
5. Assign the second process id to the first and make the status ‘not ready’.
6. Signal the handler to receive a message.
7. Get a message from the user.
8. Set the status to ‘ready’.
9. Send the message by using the kill command to interrupt the process.
10. Wait until the status is ‘not ready’ and continue.
11. Detach the pointer from the shared memory.
12. Stop

Server side:

1. Start
2. Get the process id and assign the common key value to be the key.
3. Generate the shared memory identifier.
4. Attach the structure pointer to the shared memory.
5. Set the process id with the first process and the status to ‘not ready’.
6. Signal the handler function to receive a message.
7. Wait until the status is either ‘filled’ or ‘not ready’.
8. Get a message and set the status to ‘filled’.
9. Send the message by using the kill command to interrupt the process.
10. Detach the pointer from the shared memory.

11. Destroy the shared memory space.
12. Stop

Programs:

1) Application for converting to uppercase in child:

Code:

```
//Program to implement InterProcess Communication
#include <stdio.h>
#include <stdio_ext.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

//parent writes a char in shared memory which child reads and prints in upper case
void client(int id) {
	char *a;
	a = (char*) shmat(id,NULL,0);
	printf("\nChild -> Uppercase: ");
	for (int i=0; i<strlen(a); i++)
		printf("%c",toupper(a[i]));
	printf("\n");
	shmdt((void*)a);
	}

int main() {
	int pid, id;
	char *c;

	id = shmget(IPC_PRIVATE,1024,IPC_CREAT | 00666);
	c = (char*) shmat(id,NULL,0);

	printf("Enter string: ");
	scanf("%s",c);

	pid = fork();

	printf("Parent -> %s",c);

	if (pid<0)
		printf("Error occurred");
```

```
        else if (pid==0) {
                client(id);
                exit(0);
                }


        wait(NULL);

        printf("\n");

        shmdt((void*)c);
        shmctl(id,IPC_RMID,NULL);

        return 0;
        }
```

Output:

```
kri@kri-ubuntu:~/workspace$ gcc -o q51 ipc.c
kri@kri-ubuntu:~/workspace$ ./q51
Enter string: galaxy
Parent -> galaxy
Child -> Uppercase: GALAXY
Parent -> galaxy
kri@kri-ubuntu:~/workspace$
```

2) Application for file transfer:

Code: Server side

```
//Server for file transfer application
#include <sys/ipc.h>
#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 <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

struct memory{
   char file1[30];
   char file2[30];
   char content[200];
```

```
    int status,pid1,pid2;
            };

struct memory* shmptr;

void handler(int signum) {
    if(signum==SIGUSR1){
        printf("Finding file: %s\n",shmptr->file1);

        int fd = open(shmptr->file1, O_RDONLY);


        char content[200];
        read(fd,content,200);

        strcpy(shmptr->content,content);

        shmptr->status =0;

        close(fd);

        kill(shmptr->pid2,SIGUSR2);
        exit(0);
            }
            }

int main() {

    int pid = getpid();

    int id=shmget(111,sizeof(struct memory),IPC_CREAT | 0666);
    shmptr = (struct memory*)shmat(id, NULL, 0);
    shmptr->pid1=pid;
    shmptr->status=0;

    signal(SIGUSR1, handler);

    while (1) {
        sleep(1);
            }

    shmdt((void*)shmptr);
    shmctl(id, IPC_RMID, NULL);
    return 0;
            }
```

<u>Code: Client side</u>

```
//Client for file transfer application
#include <sys/ipc.h>
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
#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 <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

struct memory {
   char file1[30];
   char file2[30];
   char content[200];
   int status,pid1,pid2;
         };

struct memory* shmptr;

void handler(int signum) {
   //printf("interupted\n");
   if(signum==SIGUSR2){
      int fd = open(shmptr->file2,O_WRONLY | O_CREAT,0644);
      write(fd,shmptr->content, strlen(shmptr->content));
      printf("%s saved as %s\n",shmptr->file1,shmptr->file2);
      exit(0);
         }
         }


int main() {
   int pid = getpid();

   int id=shmget(111,sizeof(struct memory),IPC_CREAT | 0666);
   shmptr = (struct memory*)shmat(id, NULL, 0);
   shmptr->pid2=pid;
   shmptr->status=0;

   char file1[30];
   printf("File name: ");
   scanf("%s",file1);
   strcpy(shmptr->file1,file1);

   char file2[30];
   printf("Save file as: ");
   scanf("%s",file2);
```

```
strcpy(shmptr->file2,file2);

shmptr->status=1;
kill(shmptr->pid1,SIGUSR1);

signal(SIGUSR2, handler);

while (1) {
   sleep(1);
      }

shmdt((void*)shmptr);
shmctl(id, IPC_RMID, NULL);
      }
```

Output:

```
kri@kri-ubuntu:~/workspace$ gcc -o q52c q2cl.c
kri@kri-ubuntu:~/workspace$ ./q52c
File name: foo.txt
Save file as: save.txt
foo.txt saved as save.txt
kri@kri-ubuntu:~/workspace$ cat save.txt
This is the first line.
This is the second line.
.
.
.
This is the last line.
kri@kri-ubuntu:~/workspace$
```

```
kri@kri-ubuntu:~/workspace$ gcc -o q52s q2server.c
kri@kri-ubuntu:~/workspace$ ./q52s
Finding file: foo.txt
kri@kri-ubuntu:~/workspace$ cat foo.txt
This is the first line.
This is the second line.
.
.
.
This is the last line.
kri@kri-ubuntu:~/workspace$
```

3) Chat application:

Code: Server side

```
//Server for chat application
#include <sys/ipc.h>
#include <sys/shm.h>
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
#include <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

struct memory{
   char content[200];
   int status,pid1,pid2;
         };
struct memory* shmptr;

void handler(int signum) {
   if (signum==SIGUSR1){
      printf("%32s",shmptr->content);
         }
         }

int main() {
   int pid=getpid();

   int shmid=shmget(111,sizeof(struct memory),IPC_CREAT|0666);

   shmptr=(struct memory *)shmat(shmid,NULL,0);

   shmptr->pid1=pid;
   shmptr->status=0;

   signal(SIGUSR1, handler);

   printf("\n-------------------------------\n");
   printf("\tCHAT APPLICATION\n");
   printf("-------------------------------\n");

   int ch = 1;

   do {
      fgets (shmptr->content, 200, stdin);
      if (strcmp(shmptr->content,"exit")==10) {
         ch = 0;
         break;
         }
      else
```

```
        kill(shmptr->pid2,SIGUSR2);
        } while (ch==1);

    shmdt((void*)shmptr);
    shmctl(shmid, IPC_RMID, NULL);
        }
```

Code: Client side

```
//Client for chat application
#include <sys/ipc.h>
#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 <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

struct memory {
    char content[200];
    int status,pid1,pid2;
        };
struct memory* shmptr;

void handler(int signum) {
    if (signum==SIGUSR2) {
        printf("%32s",shmptr->content);
        }
        }

int main() {

    int pid=getpid();

    int shmid=shmget(111,sizeof(struct memory),IPC_CREAT|0666);

    shmptr=(struct memory *)shmat(shmid,NULL,0);

    shmptr->pid2=pid;
    shmptr->status=0;

    signal(SIGUSR2, handler);

    printf("\n-------------------------------\n");
```

```
printf("\tCHAT APPLICATION\n");
printf("------------------------------\n");

int ch = 1;

do {
    fgets (shmptr->content, 200, stdin);
    if (strcmp(shmptr->content,"exit")==10) {
        ch = 0;
        break;
        }
    else
        kill(shmptr->pid1,SIGUSR1);
        } while (ch==1);

shmdt((void*)shmptr);
shmctl(shmid, IPC_RMID, NULL);
        }
```

Output:

```
kri@kri-ubuntu:~/workspace$ gcc -o q53c q3cl.c
kri@kri-ubuntu:~/workspace$ ./q53c

------------------------------
        CHAT APPLICATION
------------------------------
Hi!
                        Hello!
This is user1.
                 This is user2.
exit
kri@kri-ubuntu:~/workspace$
```

```
kri@kri-ubuntu:~/workspace$ gcc -o q53s q3server.c
kri@kri-ubuntu:~/workspace$ ./q53s

------------------------------
        CHAT APPLICATION
------------------------------
                           Hi!
Hello!
                 This is user1.
This is user2.
exit
kri@kri-ubuntu:~/workspace$
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- Interprocess communication was understood.
- A simulation of the process of sending and receiving signals between a client and a server was implemented.
- The concept of shared memory was understood and applied.
