Benjamin DAVID

Robin VAN DESSEL

Group 5

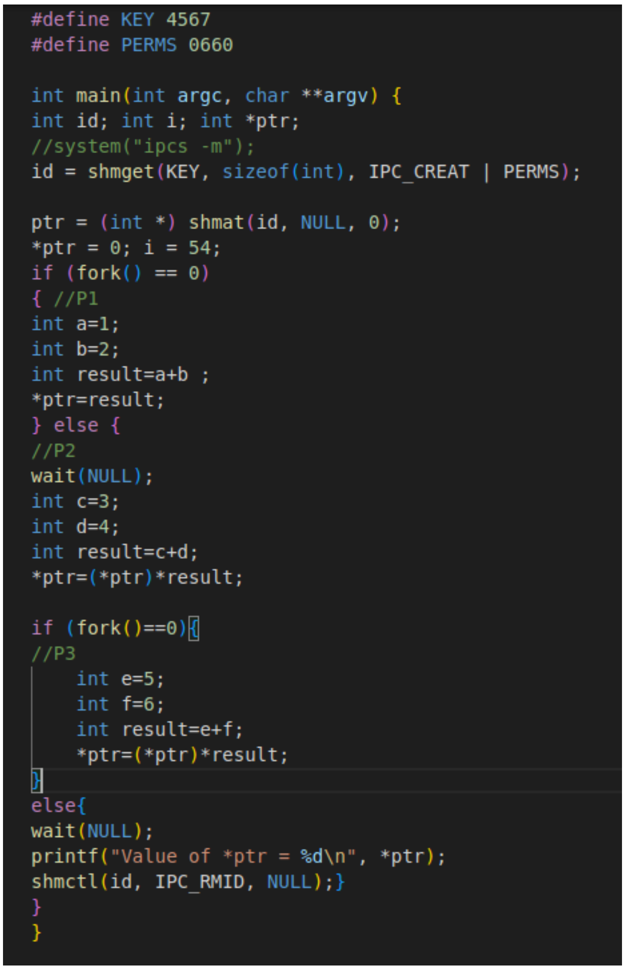
Lab 2 Report

1. Shared Memory

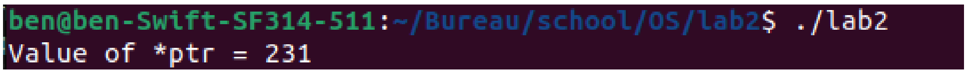
1. We can infer that *ptr is in a shared memory state. *ptr is incremented in the child process but the same incremented value is displayed by the parent and child process. It is shared through the parent and child process. Whereas ‘i’ isn’t in a shared memory space. It is only incremented in the child process and not in the parent process.

2. The shmget function creates a shared memory segment. Then, it returns in the id variable the identifier of the shared memory segment associated with the value of the key argument. The shmat function attaches the shared memory segment identified by shmid to the address space of the calling process. The need for a "KEY" here is to uniquely identify the shared memory segment.

2. Parallel Computing



```
#define KEY 4567
#define PERMS 0660

int main(int argc, char **argv) {
int id; int i; int *ptr;
//system("ipcs -m");
id = shmget(KEY, sizeof(int), IPC_CREAT | PERMS);

ptr = (int *) shmat(id, NULL, 0);
*ptr = 0; i = 54;
if (fork() == 0)
{ //P1
int a=1;
int b=2;
int result=a+b ;
*ptr=result;
} else {
//P2
wait(NULL);
int c=3;
int d=4;
int result=c+d;
*ptr=(*ptr)*result;

if (fork()==0){
//P3
    int e=5;
    int f=6;
    int result=e+f;
    *ptr=(*ptr)*result;
}
else{
wait(NULL);
printf("Value of *ptr = %d\n", *ptr);
shmctl(id, IPC_RMID, NULL);}
}
}
```



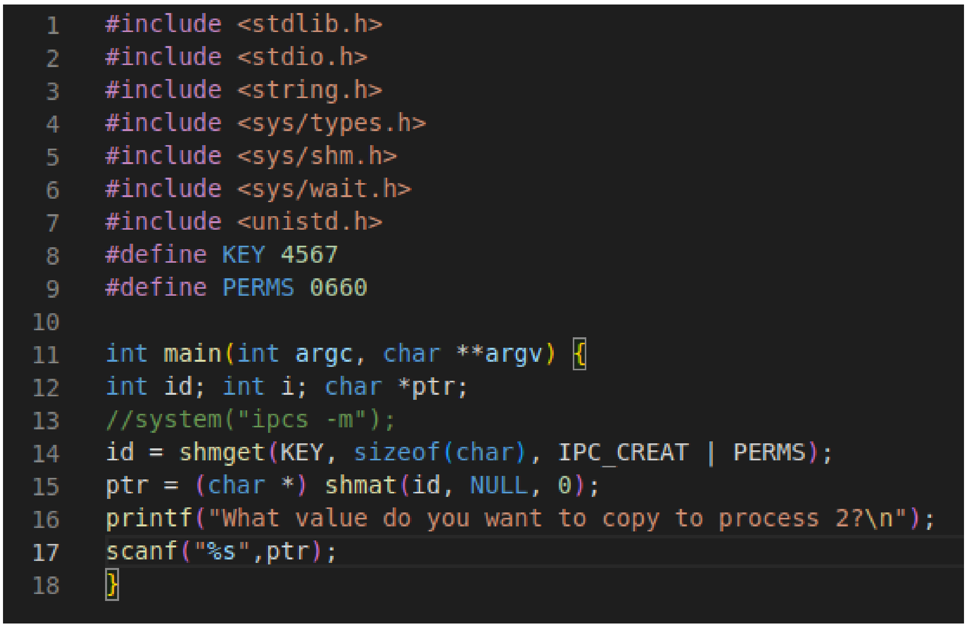
```
ben@ben-Swift-SF314-511:~/Bureau/school/OS/lab2$ ./lab2
Value of *ptr = 231
```

We first create a shared memory space with the shmget() function. Then, the first process adds a and b and put the result in the shared memory space. The second process adds c and d and add the result in the shared memory space. The third process adds e and f and add the result in the shared memory space. At the end, we print the value of ptr and close the shared memory space.

(1+2)*(3+4)*(5+6)=231.

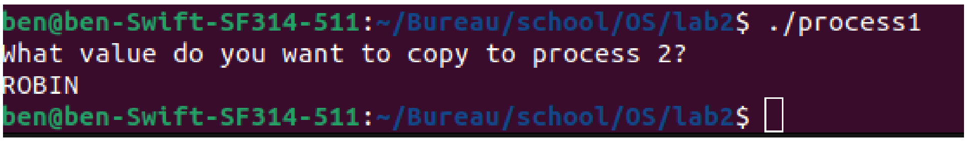
3. Implementing Copy/Paste between processes

Process 1 code:



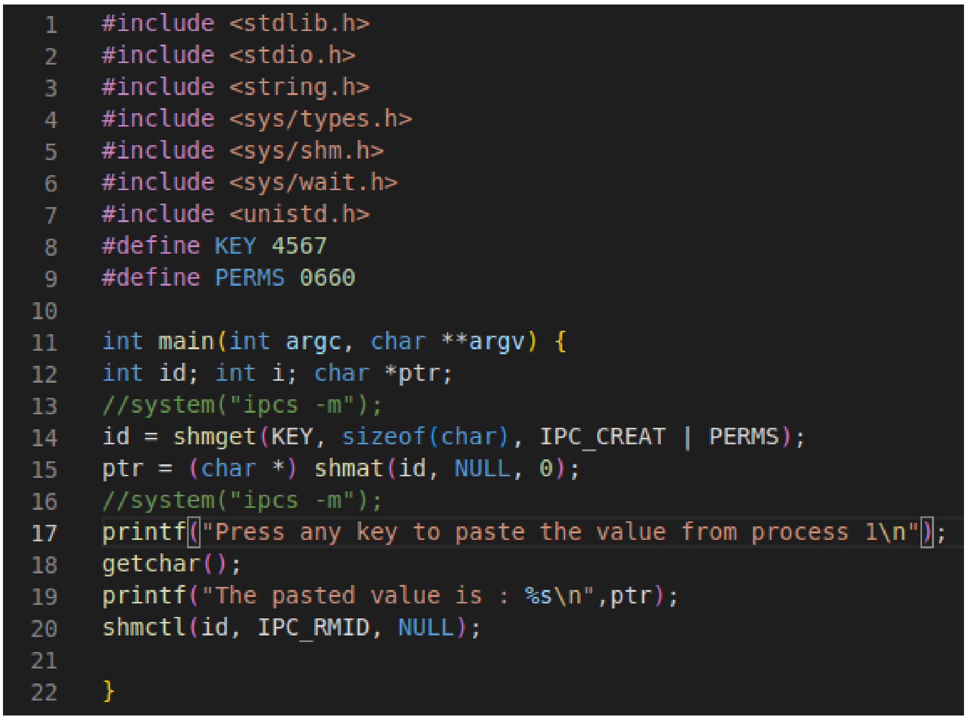
```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#define KEY 4567
#define PERMS 0660

int main(int argc, char **argv) {
int id; int i; char *ptr;
//system("ipcs -m");
id = shmget(KEY, sizeof(char), IPC_CREAT | PERMS);
ptr = (char *) shmat(id, NULL, 0);
printf("What value do you want to copy to process 2?\n");
scanf("%s",ptr);
}
```

Process 1 result:



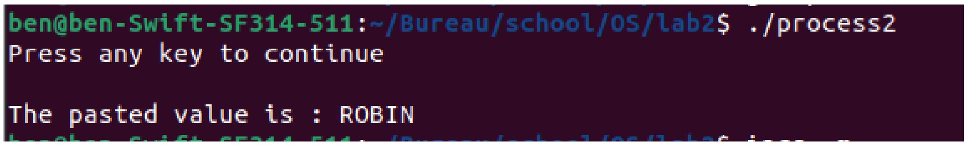
```
ben@ben-Swift-SF314-511:~/Bureau/school/OS/lab2$ ./process1
What value do you want to copy to process 2?
ROBIN
ben@ben-Swift-SF314-511:~/Bureau/school/OS/lab2$ 
```

Process 2 code:



```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#define KEY 4567
#define PERMS 0660

int main(int argc, char **argv) {
int id; int i; char *ptr;
//system("ipcs -m");
id = shmget(KEY, sizeof(char), IPC_CREAT | PERMS);
ptr = (char *) shmat(id, NULL, 0);
//system("ipcs -m");
printf("Press any key to paste the value from process 1\n");
getchar();
printf("The pasted value is : %s\n",ptr);
shmctl(id, IPC_RMID, NULL);

}
```

Process 2 result:



```
ben@ben-Swift-SF314-511:~/Bureau/school/OS/lab2$ ./process2
Press any key to continue

The pasted value is : ROBIN
ben@ben-Swift-SF314-511:~/Bureau/school/OS/lab2$ ipcs -m
```

The first process asks for an input to the user. The program first creates a shared memory space the length of a char. Then the user inputs a string. Then the second process access to the shared memory space and prints its content. It displays process 1 input. We close the space memory at the end of process 2.