DHA Suffa University

CS 206 – Operating Systems – Lab

Fall 2017

Lab 11 – Inter Process Communication

Objective(s):

- Learning about communication between processes

IPC

Inter-process communication (IPC) is a set of methods for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network.

IPC methods are divided into methods for

1. Message passing

2. Shared memory

3. Synchronization

4. Remote procedure calls (RPC).

The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

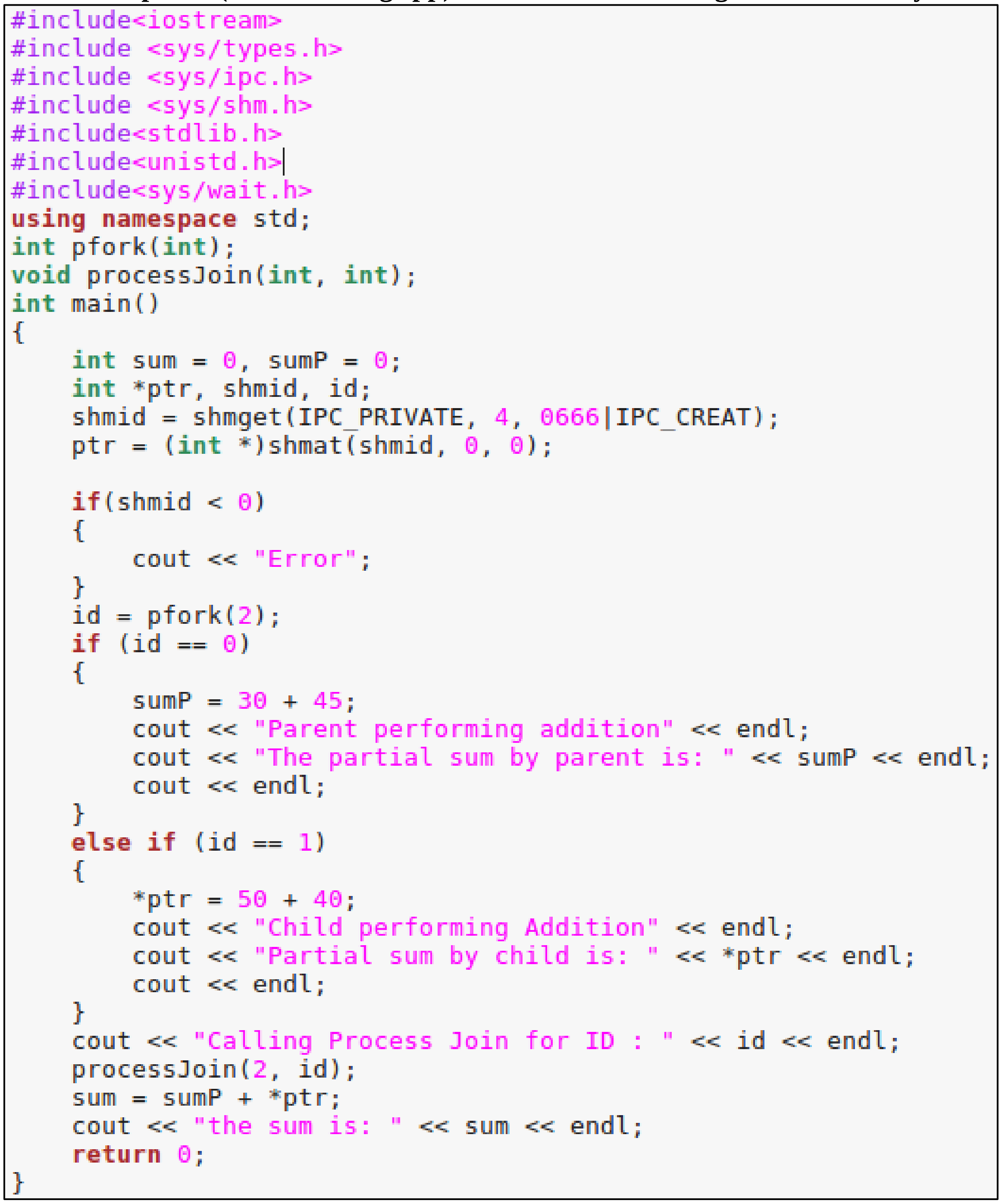
Message Passing

Example 01 (MPUsingPipe.cpp): Message Passing using Pipes

```
#include <iostream>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
using namespace std;
int main(void)
{
        int     fd[2];
        pid_t   childpid;
        char    readbuffer[80];

        pipe(fd);

        if((childpid = fork()) == -1)
        {
                cout << "Error forking";
                exit(1);
        }

        if(childpid == 0)
        {
                /* Child process closes up input side of pipe */
                close(fd[0]);

                /* Send message through the output side of pipe */
                write(fd[1], "Hello World", 11);
                exit(0);
        }
        else
        {
                /* Parent process closes up output side of pipe */
                close(fd[1]);

                /* Read in the message from the pipe */
                read(fd[0], readbuffer, sizeof(readbuffer));
                cout << "Received string: " << readbuffer << endl;
        }

        return(0);
}
```

Shared Memory Between Parent/Child Processes

Example 02 (IPCinForking.cpp): Communication using Shared Memory



```cpp
#include<iostream>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
using namespace std;
int pfork(int);
void processJoin(int, int);
int main()
{
    int sum = 0, sumP = 0;
    int *ptr, shmid, id;
    shmid = shmget(IPC_PRIVATE, 4, 0666|IPC_CREAT);
    ptr = (int *)shmat(shmid, 0, 0);

    if(shmid < 0)
    {
        cout << "Error";
    }
    id = pfork(2);
    if (id == 0)
    {
        sumP = 30 + 45;
        cout << "Parent performing addition" << endl;
        cout << "The partial sum by parent is: " << sumP << endl;
        cout << endl;
    }
    else if (id == 1)
    {
        *ptr = 50 + 40;
        cout << "Child performing Addition" << endl;
        cout << "Partial sum by child is: " << *ptr << endl;
        cout << endl;
    }
    cout << "Calling Process Join for ID : " << id << endl;
    processJoin(2, id);
    sum = sumP + *ptr;
    cout << "the sum is: " << sum << endl;
    return 0;
}
```

Continued on Next Page…

```
int pfork(int x)
{
    for(int j = 1; j <= (x-1); j++)
    {
        if(fork() == 0)
        {
            return j;
        }
    }
    return 0;
}
void processJoin(int x, int id)
{
    int val;
    if (id == 0)
    {
        for(int j = 1; j <= (x-1); j++)
        {
            cout << "Parent Process ID: " << getpid() << endl;
            cout << "Parent calling wait i.e. Blocked State" << endl;
            val = wait(0);
            cout << "The child deleted is: " << val << endl;
        }
    }
    else
    {
        cout << "Child PID: " << getpid() << endl;
        exit(0);
    }
    cout << "After exit" << endl;
}
```

Example 03 (AddArray.cpp): Adding Array Elements using Parallel Processing

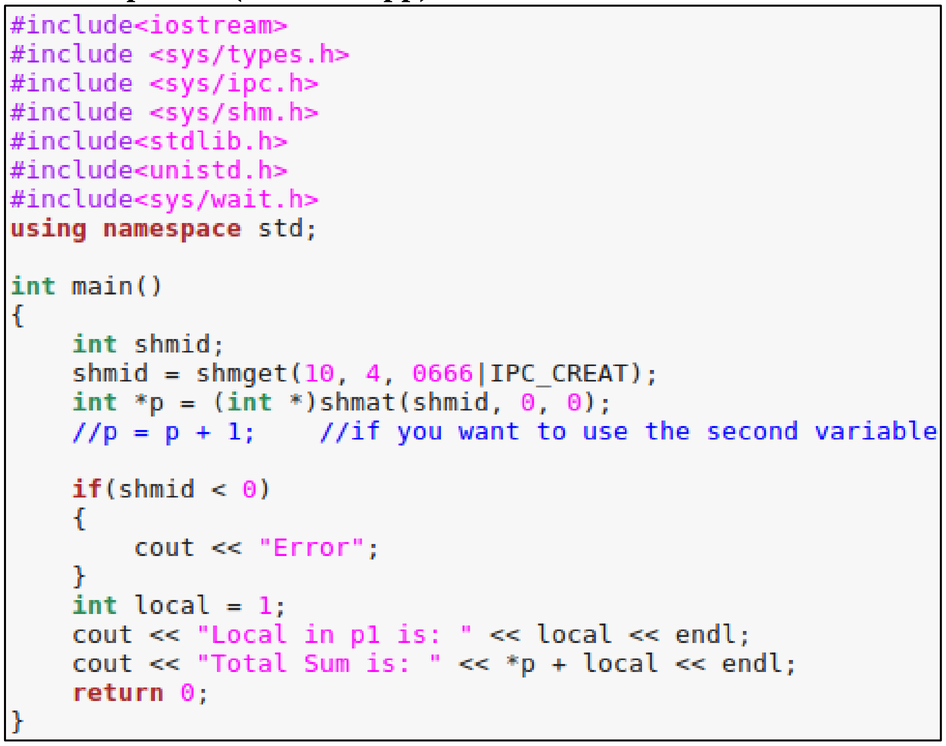
```
#include<iostream>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
using namespace std;

int main()
{
    int arr[6] = {4, 2, 6, 0, 3, 5};
    int sum = 0;
    key_t key = 4;
    int *ptr;
    int id, shmid1, shmid2;
    int pFork(int);
    void pJoin(int, int);
    shmid1 = shmget(key, 4, IPC_CREAT|0666);
    if (shmid1 < 0)
    {
        cout << "Error";
    }
    else
    {
        cout << "Share Memory ID 1 is: " << shmid1 << endl;
    }
    shmid2 = shmget(key, 4, IPC_CREAT|0666);
    cout << "Shared Memory ID 2 is: " << shmid2 << endl;
    ptr = (int *)shmat(shmid1, 0, 0);
    *ptr = 0;
    id = pFork(2);
    if (id == 0)
    {
        for (int i = 0; i < 6; i+=2)
        {
            sum = sum + arr[i];
        }
        cout << "Sum in parent is " << sum << endl;
    }
    if (id == 1)
    {
        for (int i = 1; i < 6; i+=2)
```

Continued on Next Page…

```
        for (int i = 1; i < 6; i+=2)
        {
            *ptr = *ptr + arr[i];
        }
        cout << "Sum in child is " << *ptr << endl;
    }
    pJoin(2, id);
    sum = sum + *ptr;
    cout << "The Sum is " << sum << endl;
    return 0;
}
int pFork(int x)
{
    for (int j = 1; j < x; j++)
    {
        if (fork() == 0)
        {
            return j;
        }
    }
    return 0;
}
void pJoin(int x, int id)
{
    if(id == 0)
    {
        for (int j = 1; j < x; j++)
        {
            wait(0);
        }
    }
    else
    {
        exit(0);
    }
}
```

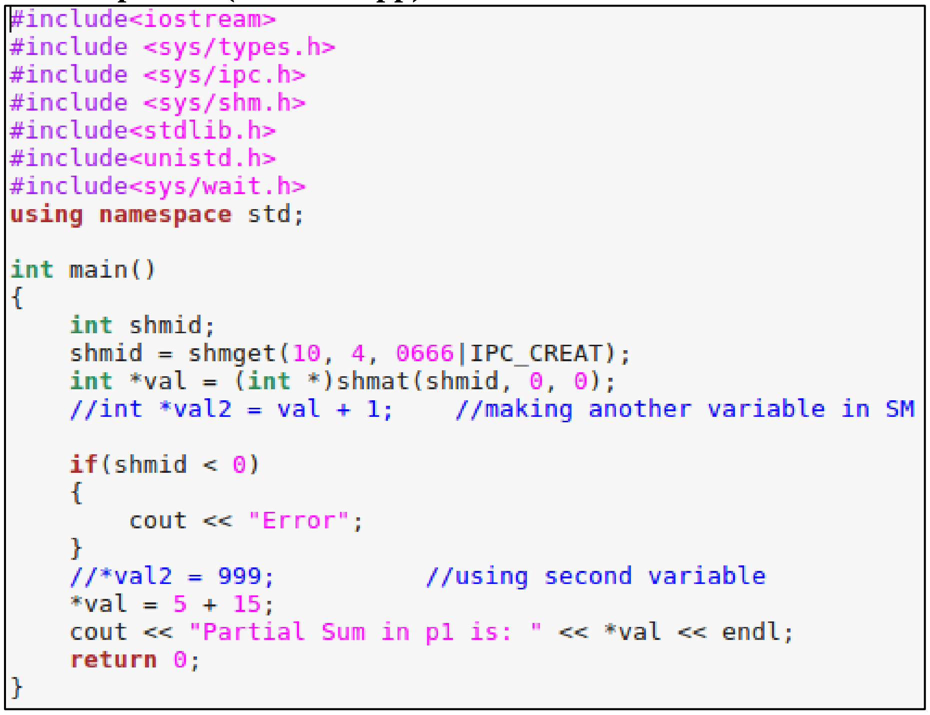
Shared Memory Between Different Processes

Example 04a (Process1.cpp): IPC Between Different Processes



```cpp
#include<iostream>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
using namespace std;

int main()
{
    int shmid;
    shmid = shmget(10, 4, 0666|IPC_CREAT);
    int *p = (int *)shmat(shmid, 0, 0);
    //p = p + 1;    //if you want to use the second variable

    if(shmid < 0)
    {
        cout << "Error";
    }
    int local = 1;
    cout << "Local in p1 is: " << local << endl;
    cout << "Total Sum is: " << *p + local << endl;
    return 0;
}
```

Example 04b (Process2.cpp): IPC Between Different Processes



```cpp
#include<iostream>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
using namespace std;

int main()
{
    int shmid;
    shmid = shmget(10, 4, 0666|IPC_CREAT);
    int *val = (int *)shmat(shmid, 0, 0);
    //int *val2 = val + 1;    //making another variable in SM

    if(shmid < 0)
    {
        cout << "Error";
    }
    //*val2 = 999;          //using second variable
    *val = 5 + 15;
    cout << "Partial Sum in p1 is: " << *val << endl;
    return 0;
}
```

Lab Task:

1. Input the weight of the user in one process and his/her height in another process. Create a variable in the memory shared between two processes which calculates, stores and prints the Body-Mass-Index of the user in either of the processes.

$$\text{BMI} = \text{weight (kg)} \div \text{height}^2 \text{ (m)}$$

Lab Assignment 11:

1. **message.cpp]** Write a program that will both send and messages and construct the following dialog between them
   (Process 1) Sends the message "Are you hearing me?"
   (Process 2) Receives the message and replies "Loud and Clear".
   (Process 1) Receives the reply and then says "I can hear you

2. **[memory.cpp**] Write a Program to create a shared memory and print the id of the shared memory
3. **[count.cpp**] Write a Program for count number of digits in string using parallel programming

Submission Instructions:

1. **Create a new folder named cs152abc where abc is your 3 digit roll #. e.g. cs152111.**
2. **Copy all the solution files into this folder.**
3. **Right-click on the folder you created and create a zip file by selecting the option**
   - **"Send to" and selecting "Compressed (zipped) folder" [for windows].**
   - **"Create Archive" and change option to ".zip" instead of ".tar.gz" and click on "Create". [for linux]**
4. **Now make sure a zip file named cs152abc.zip is created e.g. *cs152111.zip*.**
5. **Upload the assignment solution on LMS under the assignment named Lab 08 Assignment – XX, where XX is your section name.**
6. **Due Date for assignment submission is Nov 12, 2017.**