Operating Systems (CS3000)

Lecture – 15

(Inter Process Communication - 2)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING,
KANCHEEPURAM

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING,
KANCHEEPURAM

Dr. Jaishree Mayank

Assistant Professor

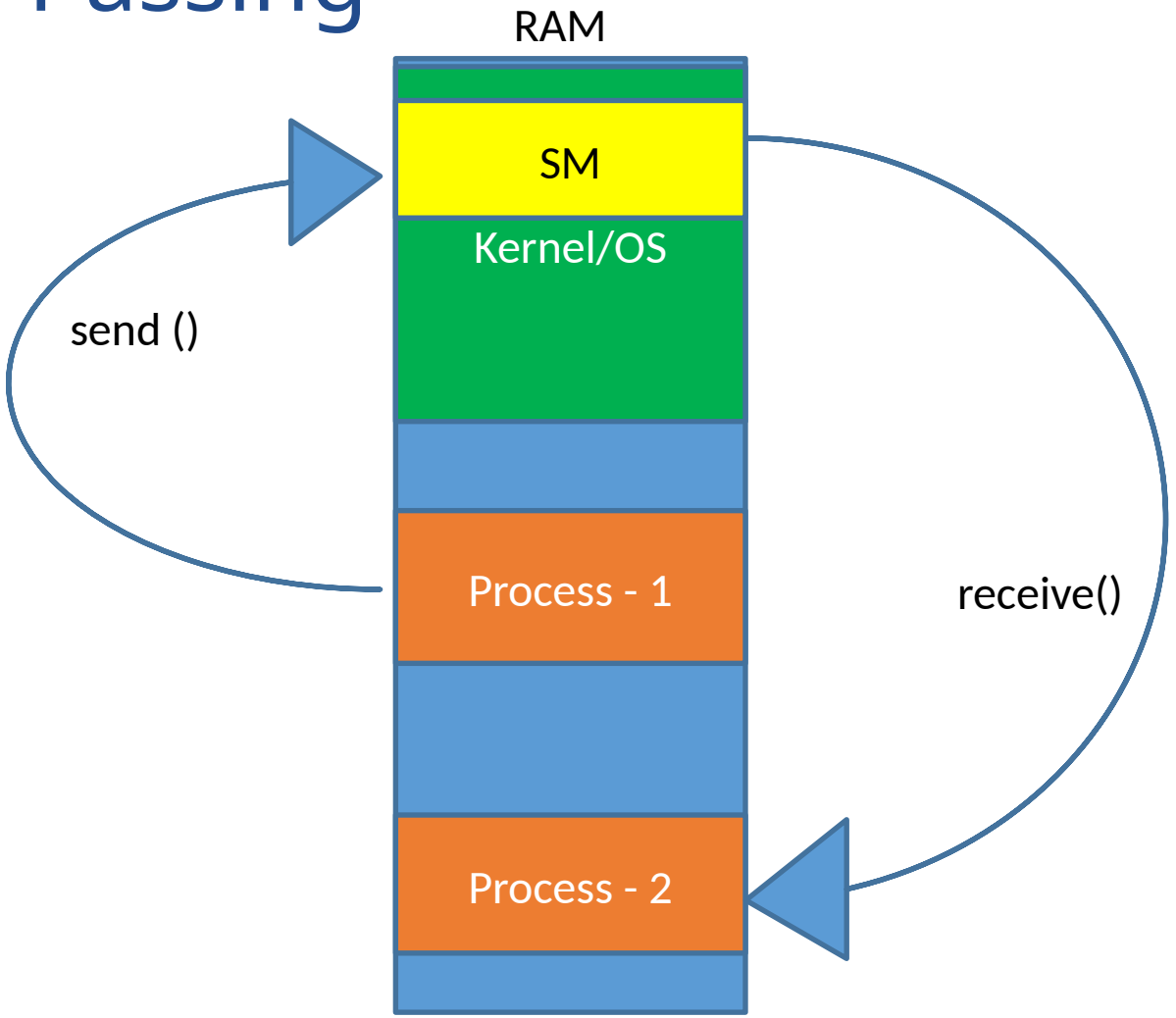
Department of Computer Sc. and Engg.

Problem with SM

- Synchronization needed between the processes

Message Passing

- SM is created in Kernel
- System calls are used
  - send(): Write to SM
  - receive(): Read from SM



IPC using Message Queues

- linked list of messages
- stored within the kernel
- identified by a message queue identifier.
- Functions
  - **msgget()** – To create a message queue/Open Existing
  - **msgsnd()** – To write message to message queue by Sender/New messages are added to the end of the queue
  - **msgrcv()** – To retrieve message from message queue by Receiver
  - **msgctl()** – The control function

IPC using Message Queues

- Sender
  - Create the MQ
  - Add data to the MQ
- Receiver
  - Retrieve the data from MQ
  - Delete the MQ

IPC using Message Queues

- int msgget(key_t key, int msgflg);
  - creates a new message queue
  - On success, msgget()returns the message queue identifier (a positive integer).
  - On failure, it returns -1and sets errnoto indicate the error.

IPC using Message Queues

- int msgsnd(int **msqid**, void ***msgp**, size_t **msgsz**, int **msgflg**);
  - send a message to the message queue specified by the msqid parameter.
  - It is returned by the msgget() function and used to identify the message queue to send the message to.
  - The *msgp parameter points to a user-defined buffer that must contain the following:
    - A field of type long int that specifies the type of the message.
    - An array that contains the actual content of the message.

IPC using Message Queues

- int msgsnd(int msqid, void *msgp, size_t msgsz, int msgflg);
  - The following structure is an example of what the user-defined buffer might look like for a message that has 5 bytes of data.

```
struct mymsg
{
        long int mtype;       /* message type */
        char mtext[5];        /* message text */
}
```

  - The value of mtype must be greater than zero. When messages are received with msgrcv(), the message type can be used to select the messages.
  - The message data can be any length up to the system limit.

IPC using Message Queues

- int msgsnd(int msqid, void *msgp, size_t msgsz, int msgflg);
-
  - msgsz: Length of the data part of the message to be sent.
  - msgflg: If the message queue is full, the msgflg parameter specifies the action to be taken. The actions are as follows:
    - 0: Suspended
    - IPC_NOWAIT: do not wait for space to become available on the message queue and return immediately.

IPC using Message Queues

- int **msgrcv**(int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);

  - The msgrcv() function reads a message from the message queue specified by the msqid parameter and places it in the user-defined buffer pointed to by the *msgp parameter.
  - The *msgp parameter points to a user-defined buffer that must contain the following:
    - A field of type long int that specifies the type of the message.
    - A data part that contains the data bytes of the message.

IPC using Message Queues

- int msgrcv(int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);

```
struct mymsg
  {
        long int mtype; /* message type */
        char mtext[5]; /* message text */
}
```

- The value of mtype is the type of the received message, as specified by the sender of the message.
- The msgsz parameter specifies the size in bytes of the data part of the message.
  - The received message is truncated to msgsz bytes if it is larger than msgsz.

IPC using Message Queues

- int msgrcv(int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);

- The msgtyp parameter specifies the type of message to receive from the message queue as follows:
  - If msgtyp = 0, read the first message in the queue.
    - the messages will the retrieved in the same order in which they were written into the message queue
  - If msgtyp > 0, the first message of type “msgtyp” is only received.
  - If msgtyp < 0, the first message of the lowest type that is less than or equal to the absolute value of msgtyp is received.

IPC using Message Queues

- int msgrcv(int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);
  - If a message of the desired type is not available on the message queue, the msgflg parameter specifies the action to be taken. The actions are as follows:
  - If the IPC_NOWAIT flag is set in the msgflg parameter, msgrcv() returns immediately with a return value of -1.
  - If 0 : suspend the process

IPC using Message Queues

- int msgctl(int msqid, int cmd, struct msqid_ds *buf);
- The msgctl() function allows the caller to control the message queue specified by the msqid parameter.
  - msqid Message queue identifier, a positive integer. It is returned by the msgget() function and used to identify the message queue on which to perform the control operation.
  - cmd Command, the control operation to perform on the message queue.
  - buf Pointer to the message queue data structure to be used to get or set message queue information.
    - msqid_ds

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/msg.h>
#define MAX_TEXT 10//Maximum length of the msg that can be sent

struct my_msg {
        long int msg_type;
        char some_text[MAX_TEXT];
};
int main()
{
        int running=1;
        int msgid;
        struct my_msg some_data;
        char buffer[10];
        msgid=msgget((key_t)3333, 0666|IPC_CREAT);
        printf("msg queue id %d\n", msgid);

        if(msgid == -1)
        {
                printf("Error in creating queue\n");
                exit(0);
        }

        while(running)
        {
                printf("Enter some text\n");
                scanf("%s", buffer);
                some_data.msg_type=1;
                strcpy(some_data.some_text, buffer);

                if(msgsnd(msgid, &some_data, MAX_TEXT, 0)==-1)
                {

                        printf("Msg not sent\n");
                }
                if(strncmp(buffer, "*",1)==0)
                {
                        running=0;
                }
        }

        return 0;
}
```

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/msg.h>
#define MAX_TEXT 10//Maximum length of the msg that can be sent

struct my_msg {
        long int msg_type;
        char some_text[MAX_TEXT];
};
int main()
{
        int running=1;
        int msgid;
        struct my_msg some_data;
        char buffer[10];
        msgid=msgget((key_t)3333, 0666|IPC_CREAT);
        printf("msg queue id %d\n", msgid);

        if(msgid == -1)
        {
                printf("Error in creating queue\n");
                exit(0);
        }
```

```c
        while(running)
        {
                printf("Enter some text\n");
                scanf("%s", buffer);
                some_data.msg_type=1;
                strcpy(some_data.some_text, buffer);

                if(msgsnd(msgid, &some_data, MAX_TEXT, 0)==-1)
                {

                        printf("Msg not sent\n");
                }
                if(strncmp(buffer, "*",1)==0)
                {
                        running=0;
                }
        }

        return 0;
}
```

Thank You

Any Questions?