**实验1. 共享内存的使用案例**

要求：实现父子进程之间通过共享内存传递消息的代码

示例：

int main() {

int shmid = shmget(1889, 1024, IPC\_CREAR|0600); //创建共享存储区

if (fork() == 0) {

 shmaddr = (char \*) shmat(shmid, NULL, 0); //建立映射

strcpy(shmaddr, “I’m child process!\n”);

shmdt(shmaddr);

return 0;

} else {

 sleep(5);

 shmaddr = (char \*) shmat(shmid, NULL, 0); //建立映射

printf(“Content of share memory: %s \n”, shmaddr);

 shmdt(shmaddr);

shmctl(shm\_id, IPC\_RMID, NULL); //共享内存不被使用，系统将自动销毁

}

}

**实验2：消息队列的使用**

要求：实现两个进程，通过一个消息队列相互发送进程信息。

示例：一个进程通过消息队列向另一个进程发送一条信息。

//发送消息的进程

#define TEXT\_SIZE 512

struct msg\_buffer{

long mtype;

char mtext[TEXT\_SIZE];

}

int main(int argc, char \* argv[]) {

int messagequeueid = -1;

struct msg\_buffer buffer;

key\_t key = 123;

//创建队列

if ( (messagequeueid = msgget(key, IPC\_CREAT)) == -1) {

 perror(“msgget error”);

 return -1;

}

//创建要传递的消息

buffer.mtype = 1;

memcpy(buffer.mtext, “A Message”, 10);

//发送消息

if(msgsnd(messagequeueid, &buffer, 10, 0) == -1) {

 perror(“fail to send message.”);

 exit(1);

}

return 0;

}

//接收消息的进程

#define TEXT\_SIZE 512

struct msg\_buffer{

long mtype;

char mtext[TEXT\_SIZE];

}

int main(int argc, char \* argv[]) {

int messagequeueid = -1;

struct msg\_buffer buffer;

memset(&buffer, 0, sizeof(buffer));

key\_t key = 123;

//通过Key获取消息队列

if ( (messagequeueid = msgget(key, IPC\_CREAT)) == -1) {

 perror(“msgget error”);

 return -1;

}

//接收消息

if(msgrcv(messagequeueid, &buffer, TEXT\_SIZE, 1, 0) == -1) {

 perror(“fail to recv message.”);

 exit(1);

}

printf(“Received message type: %ld, text: %s. \n”, buffer.mtype, buffer.mtext);

return 0;

}