[R E P O R T]

정보통신공학전공

200301582

김성태

메시지 큐를 이용한 IPC 프로그램 구현 과제 보고서

1. 과제의 목적

① 리눅스가 지원하는 프로세스간 통신 방식 중 다수의 프로세스 사이에 구조화된 데이터 블럭, 즉 메시지를 전달하는데 주로 사용되는 메시지 큐 방식에 대하여 무엇인지, 어떻게 사용하는지 공부한다.

② 공부한 내용을 점검하기 위해 기 작성된 ePDA 프로세스 관리프로그램과 sms 메시지 전송 에뮬레이션 프로그램을 활용하여 sms 전송 프로그램을 작성하고 동작을 확인해본다.

2. 필요 환경

① Linux

② GCC

3. 과제 수행 내역

① 메시지 큐에 대한 STUDY

② ePDA 프로그램 구현

4. 과제 수행

① 메시지 큐에 대한 STUDY

메세지 버퍼 (Message buffer)

```
/* message buffer for msgsnd and msgrcv calls */
/* msgsnd와 msgrcv 호출을 위한 메세지 버퍼 */
struct msgbuf {
        long mtype;     /* type of message 메세지 타입 */
        char mtext[1];  /* message text 메세지 내용 */
};
```

커널 msg 구조 (Kernel msg structure)

```
/* 시스템상에서 각 큐에 대한 msqid 구조 */
struct msqid_ds {
    struct ipc_perm msg_perm;
    struct msg *msg_first;  /* first message on queue 큐의 처음 메세지*/
    struct msg *msg_last;   /* last message in queue 큐의 마지막 메세지*/
    time_t msg_stime;       /* last msgsnd time 마지막으로 msgsnd가 수행된 시간*/
    time_t msg_rtime;       /* last msgrcv time 마지막으로 msgrcv가 수행된 시간*/
    time_t msg_ctime;       /* last change time 마지막으로 change가 수행된 시간*/
    struct wait_queue *wwait;
```

```
    struct wait_queue *rwait;
    ushort msg_cbytes;
    ushort msg_qnum;
    ushort msg_qbytes;      /* max number of bytes on queue 큐의 최대 바이트 수*/
    ushort msg_lspid;       /* pid of last msgsnd 마지막으로 msgsnd를 수행한 pid*/
    ushort msg_lrpid;       /* last receive pid 마지막으로 받은 pid*/
};
```

커널 ipc perm 구조 (Kernel ipc perm structure)

```
struct ipc_perm
{
  key_t  key;
  ushort uid;   /* owner euid and egid */
  ushort gid;
  ushort cuid;  /* creator euid and egid */
  ushort cgid;
  ushort mode;  /* access modes see mode flags below */
  ushort seq;   /* slot usage sequence number */
};
```

시스템 호출:msgget() (SYSTEM CALL:msgget())

```
  SYSTEM CALL: msgget();

  PROTOTYPE: int msgget ( key_t key, int msgflg );
    RETURNS: 성공시 메세지 큐의 확인자(message queue identifier)
             -1 on error: errno = EACCESS (접근권한이 없음)
                                  EEXIST (큐가 이미 존재하여 만들 수 없음)
                                  EIDRM (큐에 삭제 표시가 되어 있음)
                                  ENOENT (큐가 존재하지 않음)
                                  ENOMEM (큐를 만들기에 메모리가 부족함)
                                  ENOSPC (최대 큐의 갯수를 초과함)
  NOTES:
```

시스템 호출:msgsnd() (SYSTEM CALL:msgsnd())

```
SYSTEM CALL: msgsnd();

  PROTOTYPE: int msgsnd ( int msqid, struct msgbuf *msgp, int msgsz, int
msgflg );
    RETURNS: 0 on success
             -1 on error: errno = EAGAIN (queue is full, and IPC_NOWAIT was
asserted)
                                  EACCES (permission denied, no write
permission)
                                  EFAULT (msgp address isn't accessable -
invalid)
                                  EIDRM  (The message queue has been removed)
                                  EINTR  (Received a signal while waiting to
write)
                                  EINVAL (Invalid message queue identifier,
nonpositive
                                          message type, or invalid message size)
                                  ENOMEM (Not enough memory to copy message
```

```
buffer)
  NOTES:
```

시스템 호출:msgctl() (SYSTEM CALL:msgctl())

```
   SYSTEM CALL: msgctl();

  PROTOTYPE: int msgctl ( int msgqid, int cmd, struct msqid_ds *buf );
    RETURNS: 0 on success
            -1 on error: errno = EACCES (No read permission and cmd is IPC_STAT)
                                 EFAULT (Address pointed to by buf is invalid
with IPC_SET and
                                         IPC_STAT commands)
                                 EIDRM  (Queue was removed during retrieval)
                                 EINVAL (msgqid invalid, or msgsz less than 0)
                                 EPERM  (IPC_SET or IPC_RMID command was issued,
but
                                 calling process does not have write (alter)
                                 access to the queue)
  NOTES:0)
```

② ePDA 구현

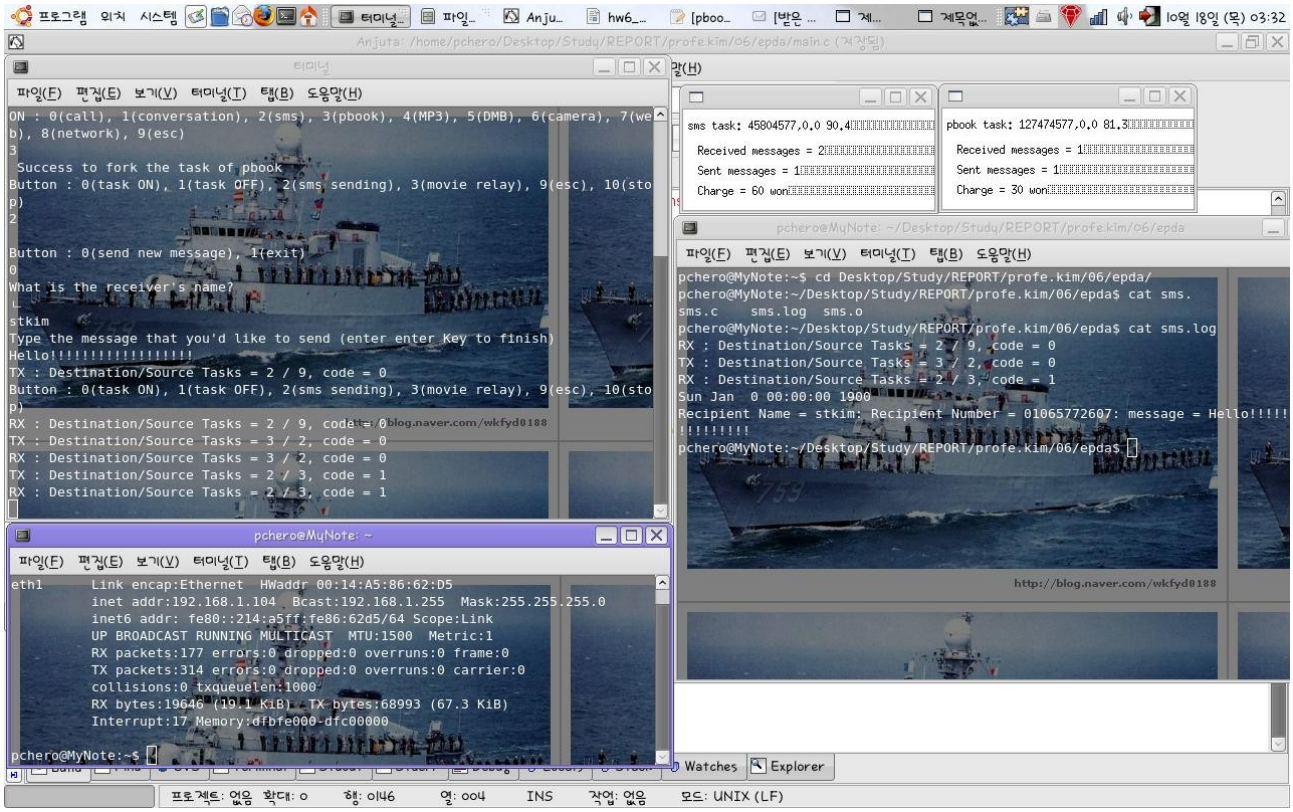

사진 *1:* 전체 실행 화면

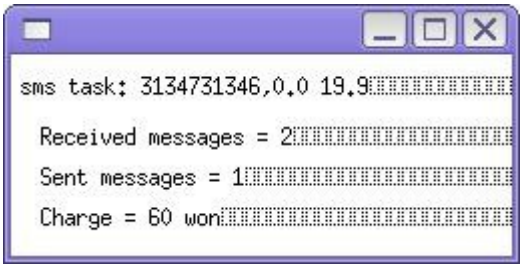

사진 *2: SMS task*

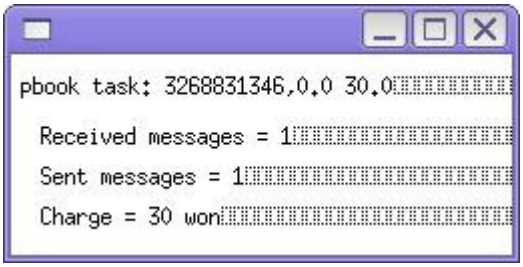

사진 *3: PBOOK task*

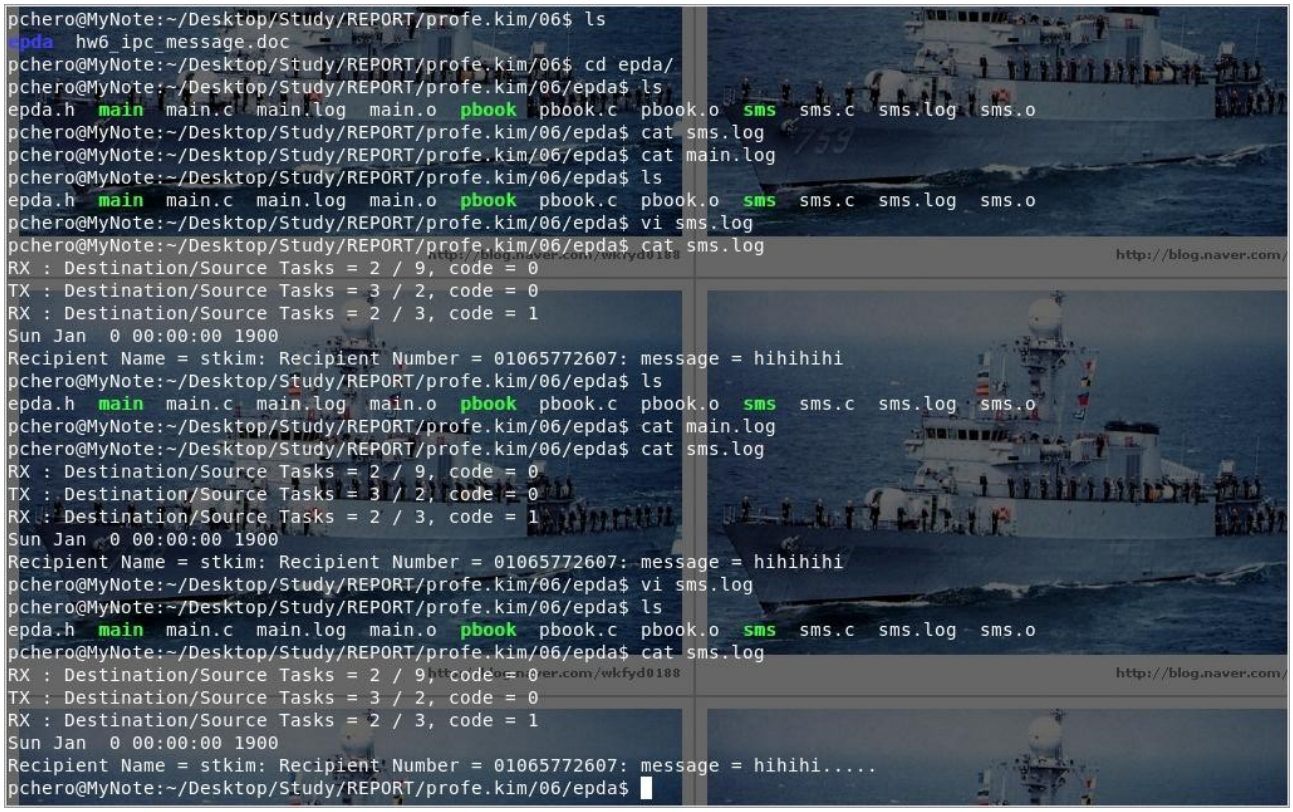

사진 *4: SMS.log file*

③ 소스파일

PBOOK.c

```
        if(rtext->SourceTaskNumber == SMS && rtext->code == REQUEST) {
                // send the recipient's telephone number request message to pbook task
                smessage.type = SMS;
                stext.SourceTaskNumber = PBOOK;
                stext.code = RESPONSE;
                strcpy(stext.RecipientName, rtext->RecipientName);
//              strcpy(mbuf, rtext->data);
//              StructCopy(smessage.text, &stext, BUFSIZ);

                // pbook task
```

```
                for(i=0;i<MAX_NUM;i++) {  //initialize phone book with null string
                        strcpy(pbook[i].num, "\0");
                        strcpy(pbook[i].name, "\0");
                }
                strcpy(pbook[MAX_NUM-1].num, "01065772607"); //insert the tjkim record at
the end of the pbook
                strcpy(pbook[MAX_NUM-1].name, "stkim");     //which is to emulate the
pbook task

                if((i = get_phone_number(stext.RecipientName)) == -1) {
                        strcpy(stext.RecipientNumber, "There is no number!\0");
                }
                strcpy(stext.RecipientNumber, pbook[i].num);

                strcpy(stext.data, rtext->data);
                StructCopy(smessage.text, &stext, BUFSIZ);



                // Send the prepared message
                if(msgsnd(mqid, (void *)&smessage, BUFSIZ, 0) == -1) {
                        fprintf(stderr, "Fail to send message!!");
                        exit(1);
                }
                MessageLog(flog, "TX", smessage);
                nsmessage++;
        }
}
```

SMS.c

```
        // if there is no message to recive, return
        if(msgrcv(mqid, (void *)&rmessage, BUFSIZ, SMS, IPC_NOWAIT) == -1)
                return;
        MessageLog(flog, "RX", rmessage);
        nrmessage++;

        rtext = (struct SMSTextFormat *) rmessage.text;

        // if receives the sms sending request from main task, request the number translation
service to pbook task
```

```
        if(rtext->SourceTaskNumber == MAIN && rtext->code == REQUEST) {
                // send the recipient's telephone number request message to pbook task
                smessage.type = PBOOK;
                stext.SourceTaskNumber = SMS;
                stext.code = REQUEST;
                strcpy(stext.RecipientName, rtext->RecipientName);
                strcpy(mbuf, rtext->data);
                StructCopy(smessage.text, &stext, BUFSIZ);

                // Send the prepared message
                if(msgsnd(mqid, (void *)&smessage, BUFSIZ, 0) == -1) {
                        fprintf(stderr, "Fail to send message!!");
                        exit(1);
                }
                MessageLog(flog, "TX", smessage);
                nsmessage++;
        }
        if(rtext->SourceTaskNumber == PBOOK && rtext->code == RESPONSE) {
                // emulate the SMS message sending by transmitting it into sms.log file
instead of radio channel
                fprintf(flog, "%s", asctime(&t));      // write calendar time in front of message.
                fprintf(flog, "Recipient Name = %s: ", rtext->RecipientName);
                fprintf(flog, "Recipient Number = %s: ", rtext->RecipientNumber);
                fprintf(flog, "message = %s\n", mbuf);
                fflush(flog);
        }
}
```

5. 느낀점

그동안 알고 있던 부분을 다시 공부하는 격이었는데...이번의 과제는 당시 공부를 했을때 대충한 부분이어서 조금 힘들었습니다. 하지만 그동안 잊고 있던 개념들을 다시금 생각해보는 좋은 계기가 되었으며, 제 자신의 부족한 부분을 돌이켜 보는 기회가 되었습니다.

6. 본인 역할

① 책을 참고하여 스스로 작성하였음.

7. 참고 사이트

① 없음.