HALLYM UNIVERSITY

CIC Convergence Information & Communications

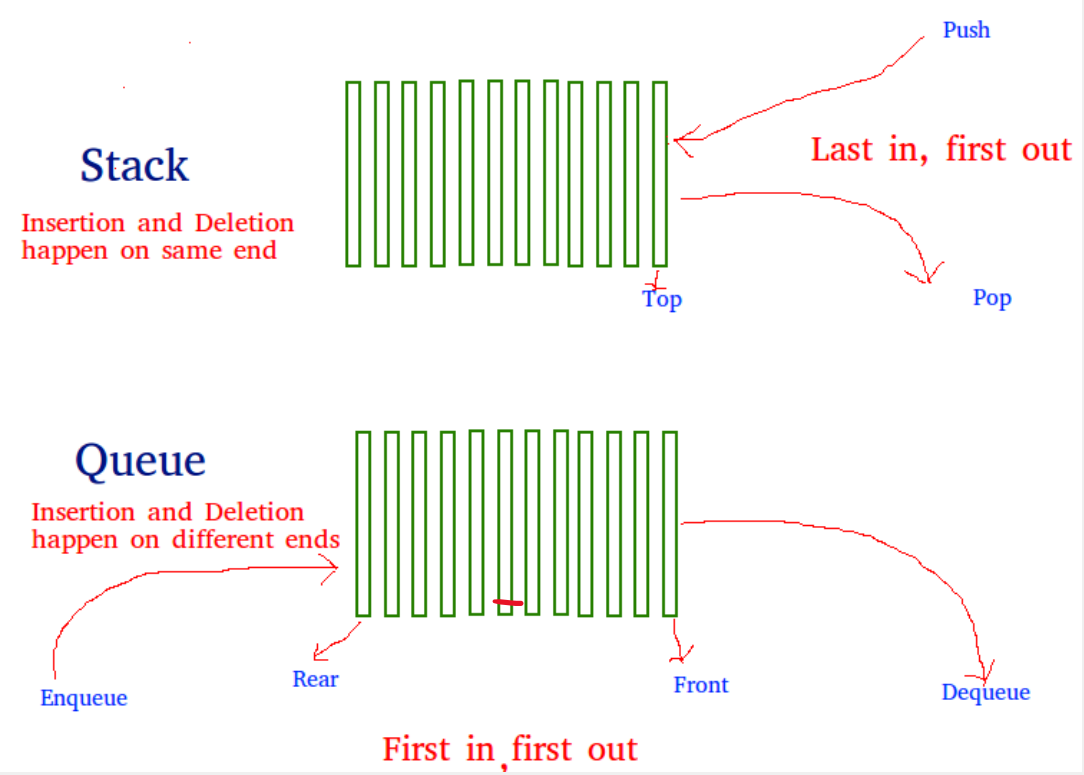
(Operating System) Practice -10-

Message Queue & Shared Memory

Index

Message Queue

- **Message Queue**
  - Message를 자료 구조의 한 가지인 queue로 관리 (즉, 선입선출)
  - 커널에서 전역적으로 관리되며, 모든 프로세스에서 접근 가능
  - 사용 방법이 매우 직관적이고 간단함



Message Queue

- **Message Queue Functions**
  - msgget → Message Queue 생성

유형	내용
헤더	#include <sys/types.h> #include <sys/ipc.h> #include <sys/msg.h>
함수	int msgget (key_t key, int msgflg);
설명	Message queue 생성
매개변수	key : 시스템에서 다른 큐와 구별되는 번호 msgflg : 옵션
반환 값	성공: message queue 식별자 실패: -1

  - msgget의 동작 옵션 (즉, msgflag)

msgflg	내용
IPC_CREAT	key에 해당하는 큐가 있다면 큐의 식별자를 반환하고, 없다면 큐를 생성
IPC_EXCL	key에 해당하는 큐가 없다면 큐를 생성하고, 큐가 있다면 -1을 반환

Message Queue

- **Message Queue Functions**
  - msgsnd → Message Queue에 메시지 전송
    - ✓ 프로세스로부터 연속적으로 수신되는 메시지는 연결 리스트로 계속해서 저장됨

유형	내용
헤더	#include <sys/types.h> #include <sys/ipc.h> #include <sys/msg.h>
함수	int msgsnd (int msqid, const void * msgp, size_t msgsz, int msgflg);
설명	Message queue에 message 전송
매개변수	msqid : message queue 식별자 msgp : 전송할 자료 msgsz : 전송할 자료의 크기 msgflg : 동작 옵션
반환 값	성공: 0 실패: -1

  - msgsnd의 동작 옵션 (즉, msgflag)

msgflg	내용
0	큐에 공간이 생길 때까지 대기
IPC_NOWAIT	큐에 여유 공간이 없다면 바로 -1로 복귀

Message Queue

- **Message Queue Functions**
  - msgrcv → Message Queue로 부터 메시지 수신

유형	내용
헤더	#include <sys/types.h> #include <sys/ipc.h> #include <sys/msg.h>
함수	int msgrcv (int msqid, const void * msgp, size_t msgsz, long msgtyp, int msgflg);
설명	Message 수신
매개변수	msqid : message queue 식별자 msgp : 전송 받을 자료 msgsz : 전송 받을 자료의 크기 Msgtyp: 전송 받을 데이터의 종류 msgflg : 동작 옵션
반환 값	성공: 0 실패: -1

Message Queue

- **Message Queue Functions**
  - msgrcv의 동작옵션 (즉, msgtyp & msgflg)

msgtyp	내용
0	큐에 자료가 있다면 첫 번째 자료를 읽어 들임
양수	양수로 지정한 값과 같은 data_type의 자료 중 첫 번째 자료를 읽어 들임
음수	음수 값을 절대값으로 변경하고, 이 절대값과 같거나 보다 제일 작은 data_type의 자료를 읽어 들임 message queue에 data_type이 1, 5, 15이고, -10을 지정했다면 1의 데이터를 읽어 들임

msgflg	내용
IPC_NOWAIT	큐에 자료가 없다면 기다리지 않고 -1로 복귀
MSG_NOERROR	큐에 있는 자료가 준비된 크기보다 크다면, 초과되는 부분을 잘라내고 읽어 들일 수 있는 부분만 가져옴 이 옵션이 없을 경우, 큐에 자료가 있더라도 -1로 실패함

Message Queue

- **Message Queue Functions**
  - msgctl → Message Queue 상태 제어

유형	내용
헤더	#include <sys/types.h> #include <sys/ipc.h> #include <sys/msg.h>
함수	int msgctl (int msqid, int cmd, struct msqid_ds * buf);
설명	Message queue state control
매개변수	msqid : message queue 식별자 cmd : 제어 명령 buf : message queue 자료를 받을 버퍼
반환 값	성공: 0 실패: -1

  - msgsnd의 동작 옵션 (즉, cmd)

cmd	내용
IPC_STAT	큐의 현재 상태를 buf에 저장
IPC_SET	큐의 상태를 buf 값으로 변경 모든 정보는 저장할 수 없으며, msg_perm과 msg_qbytes 내용만 변경 가능
IPC_RMID	큐를 삭제함 큐 삭제 시, 버퍼가 필요 없으므로 buf를 0으로 지정

Message Queue Practice -1

- **Message Queue Practice**

msq_sender.c (1/2)

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include <string.h>

struct personal_data{
    char age[16];
    char name[16];
};
struct message{
    long msg_type;
    struct personal_data data;
};

void printMsgInfo(int msqid){
    struct msqid_ds m_stat;
    printf("===== message queue info =====\n");
    if(msgctl(msqid,IPC_STAT,&m_stat)==-1){
        printf("msgctl failed\n");
        exit(0);
    }
    printf(" message queue info\n");
    printf(" msg_lspid: %d\n", m_stat.msg_lspid);
    printf(" msg_qnum: %ld\n", m_stat.msg_qnum);
    printf(" msg_stime: %ld\n", m_stat.msg_stime);
    printf("============================\n");
}
```

Message Queue Practice -1

- **Message Queue Practice**

msq_sender.c (2/2)

```
31  void main(){
32      key_t key = 12345;
33      int msqid;
34      struct message msg;
35      msg.msg_type = 1;
36      printf("What is your name?: ");
37      fgets(msg.data.name, sizeof(msg.data.name), stdin);
38      printf("What is your age?: ");
39      fgets(msg.data.age, sizeof(msg.data.age), stdin);
40
41      //GET msqid
42      if(msqid=msgget(key,IPC_CREAT|0666)==-1){
43          printf("msgget failed\n");
44          exit(0);
45      }
46
47      //Check msqid_ds before sending message
48      printMsgInfo(msqid);
49
50      //Sending message
51      if(msgsnd(msqid, &msg, sizeof(struct personal_data), 0)==-1){
52          printf("msgsnd failed\n");
53          exit(0);
54      }
55      printf("message has been sent.\n");
56      printMsgInfo(msqid);
57  }
```

Message Queue Practice

- **Message Queue Practice**

msq_receiver.c (1/2)

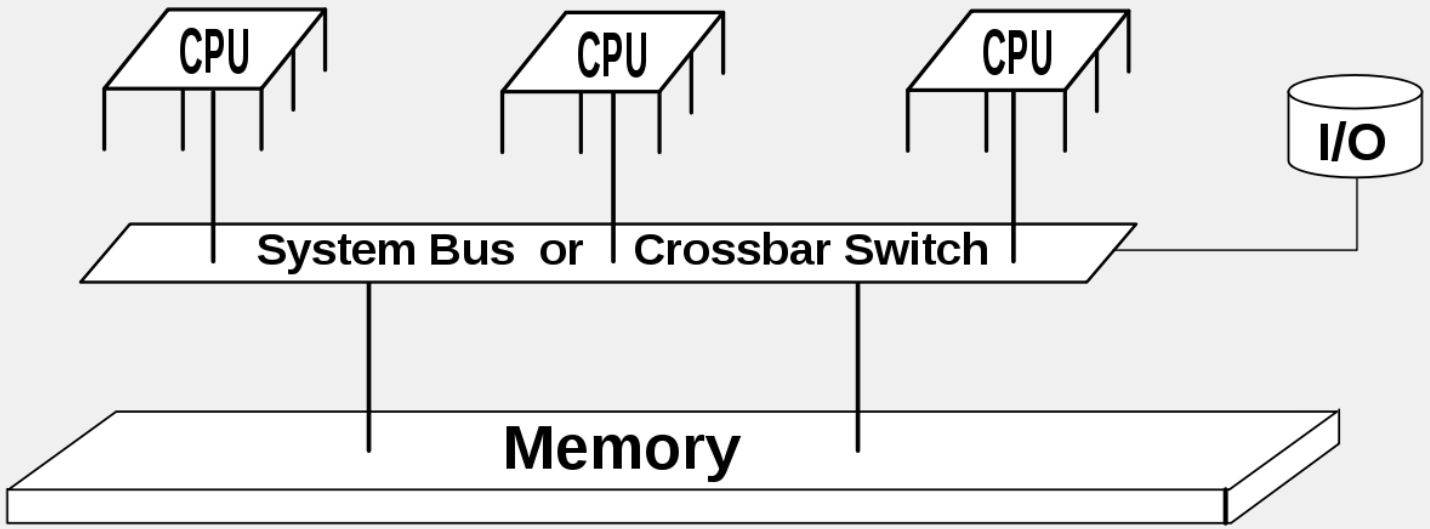
```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include <string.h>

struct personal_data{
    char age[16];
    char name[16];
};
struct message{
    long msg_type;
    struct personal_data data;
};
```

msq_receiver.c (2/2)

```c
void main(){
    key_t key = 12345;
    int msqid;
    struct message msg;

    //Get msqid
    if(msqid=msgget(key,IPC_CREAT|0666)==-1){
        printf("msgget failed\n");
        exit(0);
    }
    //Receive message
    if(msgrcv(msqid,&msg, sizeof(struct personal_data),0,0)==-1){
        printf("msgrcv failed\n");
        exit(0);
    }

    printf("name: %s, age: %s\n", msg.data.name,msg.data.age);

    if(msgctl(msqid,IPC_RMID,NULL)==-1){
        printf("msgctl failed\n");
        exit(0);
    }
}
```

Shared Memory

- **Shared Memory**
  - 일반적으로 메모리 공간은 하나의 프로세스가 점유하여 사용
  - 그러나, Shared Memory를 사용하여, 여러 프로세스가 메모리 공간을 공유할 수 있음
  - Shared Memory를 통해 과다한 데이터의 복사를 피하고, 프로세스간 통신을 수행할 수 있음



Shared Memory

- **Shared Memory Functions**
  - shmget → 공유 메모리 할당

유형	내용
헤더	#include <sys/ipc.h> #include <sys/shm.h>
함수	int shmget(key_t key, size_t size, int shmflg);
설명	key의 값을 입력하여 공유메모리를 할당함. 공유메모리 조각(shared memory segment)의 id를 반환함.
매개변수	key : 공유메모리를 할당할 때 사용하는 고유 key 값. size_t: 메모리의 최소 size. 만약 이미 존재하는 메모리라면 0으로 표기함. shmflg: 옵션 플래그
반환 값	성공: shared memory id식별자 실패: -1

  - shmflg의 동작 옵션 (즉, msgflag)

shmflg	내용
IPC_CREAT	새로운 메모리 세그먼트를 생성 (덮어쓰기)
IPC_EXCL	만약 기존 공유 메모리 세그먼트가 존재하면 shmget은 실패 (-1 반환)

Shared Memory

- **Shared Memory Functions**
  - shmat → 현재 프로세스와 공유 메모리와 연결(attach)

유형	내용
헤더	#include <sys/ipc.h> #include <sys/shm.h>
함수	void *shmat(int shmid, const void *shmaddr, int shmflg);
설명	주어진 공유 메모리 id에 해당하는 공유메모리와 프로세스를 연결함.
매개변수	shmid: 공유 메모리의 id. shmaddr: NULL(0) → 공유메모리의 주소를 반환. shmflg: 옵션 플래그
반환 값	성공: **void pointer** 실패: -1

Shared Memory

- **Shared Memory Functions**
  - shmdt → 현재 프로세스와 공유 메모리와 연결해제(detach)

유형	내용
헤더	#include <sys/ipc.h> #include <sys/shm.h>
함수	int shmdt(const void *shmaddr);
설명	주어진 공유 메모리 id에 해당하는 공유메모리와 프로세스를 연결함.
매개변수	shmid: **shmat**에서 전달받은 void 포인터
반환 값	성공: 0 실패: -1

Shared Memory

- **Shared Memory Functions**
  - shmctl → 공유 메모리 제어 (삭제, 정보조회 등)

유형	내용
헤더	#include <sys/ipc.h> #include <sys/shm.h>
함수	int shmctl(int shmid, int cmd, struct shmid_ds *buf);
설명	공유메모리를 제어함 (삭제, 정보 조회 등)
매개변수	shmid: shmat에서 전달받은 그 포인터 cmd : 명령어 → 정수형을 갖으며 여러 command를 통해 다양한 작업 수행 (삭제, 정보 조회 등) buf : shmid_ds라는 구조체로 정의되어 있음. 공유메모리의 정보를 담는 구조체
반환 값	성공: 0 실패: -1

Shared Memory Practice

- **Shared Memory Practice**

shm_writer.c (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <signal.h>

#define SIZE 1024

void signalHandler(int signum);
int shmid;
```

shm_writer.c (2/2)

```c
void main (){
    void *shmaddr; // void pointer
    shmid = shmget((key_t)1234, SIZE, IPC_CREAT|0666);
    if (shmid==-1){
        printf("shmget failed\n");
        exit(0);
    }
    shmaddr = shmat(shmid, (void *)0, 0);
    if(shmaddr==(void *)-1){
        printf("shmat error\n");
        exit(0);
    }

    printf("Writing Hello message in the shared memory.\n");
    strcpy((char*)shmaddr, "Hello there~?"); // save data in the shared memory
    if(shmdt(shmaddr)==-1){
        printf("shmdt error\n");
        exit(0);
    }

    //set action for signal
    signal(SIGINT, signalHandler);
    //waiting for signal
    pause();
}

void signalHandler(int signum){
    if(shmctl(shmid, IPC_RMID, 0)==-1){
        printf("shmctl error\n");
        exit(1);
    }
    printf("========================\n");
    printf("Got SIGINT, it will ends\n");
    exit(0);
}
```

Shared Memory Practice

- **Shared Memory Practice**

shm_reader.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <signal.h>

#define SIZE 1024

void main (){
    int shmid;
    void *shmaddr;
    struct shmid_ds shm_stat;
    //shmid_ds defined in sys/shm.h & includes the info of shared memory

    shmid = shmget((key_t)1234, SIZE, IPC_CREAT|0666);
    if (shmid==-1){
        printf("shmget failed\n");
        exit(0);
    }

    shmaddr = shmat(shmid, (void*)0, 0);
    if(shmaddr==(void *)-1){
        printf("shmat failed\n");
        exit(0);
    }

    printf("read data from shared memory as follows: %s\n", (char *)shmaddr);

    if(shmctl(shmid,IPC_STAT,&shm_stat)==-1){
        printf("shmdt error\n");
        exit(0);
    }
    printf("=====================================================\n");
    printf("Sending SIGINT to the creator of the shared memory.\n");
    kill(shm_stat.shm_cpid, SIGINT);
}
```