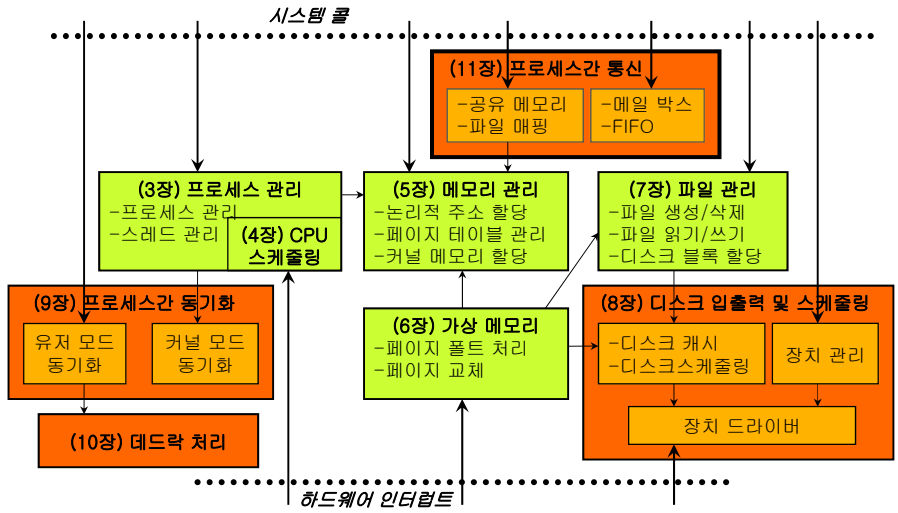
프로세스 간의 통신

- ✓공유 메모리 방식과 메시지 전달 방식
- ✓공유 메모리와 파일 매핑
- ✓메일 박스와 랑데부
- ✓파이프와 스트림 소켓
- ✓다자간 통신
- ✓유닉스 시그널
- ✓여러가지 고려사항들
- ✓picoKernel의 메일박스/랑데부/시그널 구현

관련 운영체제 구성 모듈

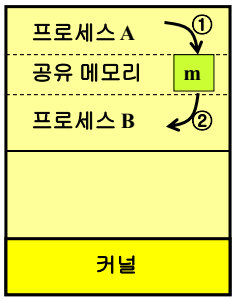


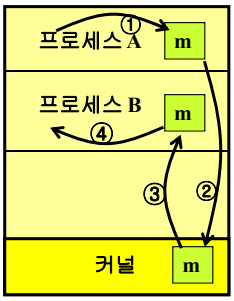
이해하고 넘어가야 할 내용들

- ✓ 공유 메모리 방식과 메시지 전달방식의 차이점 및 장단점에 대한 이해
- ✓ 공유 메모리와 파일매핑의 이해 및 이들을 메모리 관리 및 가상메모리 기능을 활용하여 구현하는 방법에 대한 이해
- ✓ 메시지 전달 방식으로서 메일박스, 랑데부, 파이프 등의 구현 방법에 대한 이해
- ✓ 유닉스 시그널 기능의 용도와 구현방법에 대한 이해

프로세스 간의 통신

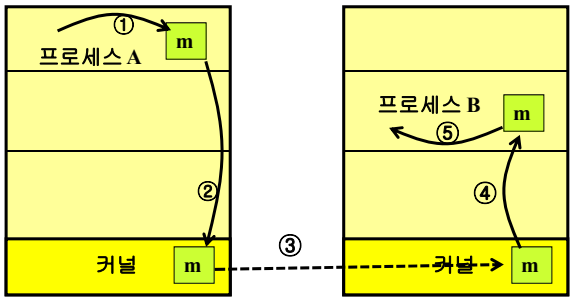
- ✓ **프로세스 간의 통신 (IPC: Inter-Process Communication)**
  - ✓ 병행하여 실행하는 프로세스들 간에 데이터 교환
  - ✓ 클라이언트-서버 프로세스 간에 요구정보 전달 및 결과 반송
  - ✓ 공유 메모리 (shared memory)방식과 메시지 전달 (message passing) 방식
- ✓ **공유 메모리 방식** **메시지 전달 방식**





원격 컴퓨터상의 프로세스간

- ✓ 메시지 전달 방식
  - ✓ 인터넷 소켓 프로그래밍



공유메모리

- ✓ 유한버퍼 문제에 공유메모리를 적용한 예
  - ✓ 프로그램 11.1
  - ✓ 동기화를 위해 2개의 세마포 사용
- ✓ 스레드 간의 공유메모리 적용
  - ✓ 동일한 프로세스에 소속된 스레드 간에는 모든 전역변수와 추가 할당받은 메모리를 공유

  ← 하나의 논리적 주소공간을 공유, 하나의 페이지 테이블
- ✓ 프로세스간의 공유메모리 적용
  - ✓ 별도로 공유메모리를 생성하고 프로세스 별로 등록하는 절차 필요
  - ✓ POSIX 표준: shmget (생성), shmat (등록, attach), shmdt (제거, detach)
  - ✓ 커널은 주소등록 요청시 프로세스의 빈 논리적 주소공간을 할당하고 페이지 테이블에 등록

  → 같은 물리적 주소를 각각 자신의 논리적 주소영역에 등록

파일 매핑

- ✓ **파일 매핑 (file mapping 또는 memory-mapped file)**
  - ✓ 파일에 기록하는 작업을 메모리 영역을 읽고 쓰는 작업으로 처리
  - ✓ 파일 매핑 등록 절차 필요
  - ✓ POSIX: mmap(), munmap()
  - ✓ Windows: CreateFileMapping(), MapViewOfFile(), UnmapViewOfFile()
- ✓ **공유 메모리 기능으로 활용**
  - ✓ 프로세스들이 동일한 파일을 파일 매핑으로 등록
- ✓ **파일 매핑의 구현**
  - ✓ 가상메모리 기능을 활용하여, 파일크기 만큼의 논리적 주소공간을 확보하고 대상 파일을 스왑 디바이스로 처리
  - ← 공유메모리 이되 대상파일을 스왑 디바이스로 지정하는 것임
  - ← unmap 시점에 파일에 기록 완료

메일박스와 랑데부

- ✓ **메일박스 (mail box)**
  - ✓ 대표적인 메시지 전달 방식 기능
  - ✓ 송신자: 메일박스에 전송 요청 (mboxSend / msgsnd)
  - ✓ 수신자: 메일박스에서 수신 요청 (mboxReceive / msgrcv)
  - ✓ 별도의 메일박스 생성절차 필요 (mboxCreate / msgget)
  - ✓ 구현방법: 커널 내에 순환버퍼 형태로 관리하고 송신자와 수신자간에 적절한 동기화 처리
  - ✓ ㈜ 메시지 큐, 메시지 포트, 메일 슬롯 등 운영체제마다 약간씩 다른 기능을 제공함
- ✓ **랑데부 (rendezvous)**
  - ✓ 송신자와 수신자가 모두 준비되었을 때 송신자의 버퍼로부터 수신자의 버퍼로 직접 복사
  - ✓ 메일박스에서 커널 버퍼의 크기가 0 인 경우에 해당

파이프와 스트림 소켓

- ✓ **바이트 스트림의 전송**
  - ✓ 파이프 (pipe): 송신자의 표준출력을 수신자의 표준입력으로 연결
  - ✓ FIFO (또는 named pipe): 파일시스템 상에 FIFO 타입의 파일을 생성하고 여기에 쓰기 및 읽기로 송신 및 수신 처리
  - ✓ 스트림 소켓 (stream socket): 스트림 타입의 소켓을 생성하고 여기에 쓰기 및 읽기로 송신 및 수신 처리
  - ✓ 구현방법: 커널 내에 순환버퍼 형태로 관리하고 송신자와 수신자간에 적절한 동기화 처리
- ✓ **유닉스의 구현**
  - ✓ 파이프: 시스템 콜 함수 pipe 로 생성, 부모-자식 프로세스 간에 사용
  - ✓ FIFO: 시스템 콜 함수 mkfifo 로 생성, 파일 이름으로 구별
  - ✓ 소켓: 시스템 콜 함수 socket 으로 생성, 포트 번호로 구별

다자간 통신

- ✓ **다자간 통신**
  - ✓ 다수 송신자와 하나의 수신자
  - ✓ 다수의 송신자와 다수의 수신자
  - ✓ 하나의 송신자와 다수의 수신자 (방송, 멀티캐스트)
- ✓ **메일박스를 이용한 다자간 통신**
  - ✓ 동일한 메일박스에 전송 및 수신을 요청
  - ✓ 경매문제 프로그램 예: 다수 송신자와 하나의 수신자 (프로그램 11.3)
- ✓ **공유 메모리를 이용한 다자간 통신**
  - ✓ 유한버퍼문제의 버퍼수정 부분을 크리티컬 섹션으로 처리
- ✓ **공유 메모리를 이용한 방송/멀티캐스트**
  - ✓ 송신자 프로세스는 공유메모리에 기록
  - ✓ 모든 수신자는 공유메모리에서 읽기
  - ✓ 모든 수신자가 다 읽었는지를 확인할 수 있는 동기화 필요

메일박스를 이용한 다자간 통신

```
struct bidMessage {
  int bidder;
  int price;
};

int server(int mboxid)
{
  int CurrentPrice = 0;
  struct bidMessage bid;
  while (1) {
      mboxReceive(mboxid, &bid);
      if (bid.price > CurrentPrice) {
        CurrentPrice = bid.price;
        printf("current price is %d\n", CurrentPrice);
      }
  }
}
```


메일박스를 이용한 다자간 통신

```
int bidder(int mboxid)
{
  struct bidMessage mybid;
  mybid.bidder = threadSelf();
  while (1) {
      mybid.price = select bidding price;
      mboxSend(mboxid, &mybid);
  }
}
int userMain(int arg)
{
  int mboxid;
  mboxid = mboxCreate();
  threadCreate(server, 20, mboxid);
  threadCreate(bidder, 20, mboxid);
  ...
}
```

유닉스 시그널

- ✓ **유닉스의 시그널 (signal)**
  - ✓ 프로세스 간의 신호 전송
  - ✓ 프로세스의 정지/강제종료, 알람, 자식/부모 프로세스의 종료, 다른 프로세스의 강제종료 등
  - ✓ 시그널별로 수신시 실행할 함수 (handler)를 사전에 등록
  - ✓ 프로세스에 시그널 발생은 CPU에 인터럽트 발생과 같은 개념
  - ✓ 시그널 핸들러는 인터럽트 처리 루틴과 같은 개념
- ✓ **시그널의 사용**
  - ✓ 핸들러 함수의 등록: 시스템 콜 함수 signal 또는 sigaction
  - ✓ 시그널의 전송: 시스템 콜 함수 kill
- ✓ **시그널과 핸들러의 사용 예**
  - ✓ 프로그램 11.4 (picoKernel의 시그널 적용)

여러가지 고려 사항들

- ✓ **통신방식의 선택**
  - ✓ 운영체제 설계시 목적에 맞게 포함할 기능 선택
  - ✓ 응용 프로그램 작성시 운영체제에서 제공하는 기능중 적절한 것 선택
- ✓ **운영체제 설계 / 응용 프로그램 작성**
  - ✓ 동기식 (synchronous) 과 비동기식 (asynchronous)
  - ✓ 직접 통신과 간접 통신
  - ✓ 단방향 (unidirection) 과 양방향 (bidirection)
  - ✓ 메시지 크기의 가변성
- ✓ **운영체제 설계**
  - ✓ 네이밍 (naming): 통신 채널을 구별하여 지정하는 방법 (고유번호, 파일이름 등)
- ✓ **응용 프로그램 작성시 고려사항**
  - ✓ 데이터 양, 프로그램 작성의 편리성, …

picoKernel의 메일박스/랑데부/시그널

- ✓ 메일박스의 구현 (프로그램 11.5)
- ✓ 랑데부의 구현 (프로그램 11.6)
- ✓ 시그널의 구현 (프로그램 11.7)