**2019062833 컴퓨터소프트웨어 학부 김유진**

**운영 체제 HW#3**

**제출 일자: 2021/03/28**

1. **과제 A**

* **Named Pipe**

1. **프로그램 설명**

**1.자료구조**

**1)파이프: 단방향 통신으로 부모 자식 간의 프로세스 통신에 사용.**

**2)named 파이프: 파이프에 이름을 붙여서 부모 자식이 아닌 임의의 프로세서 사이에서의 통신이 가능하게 함**

**3)FIFO 특수 파일: named 파이프의 역할을 대신함.**

**2.함수**

**1)access: 해당 파이프를 참조중인 수를 반환**

**2)mkfifo: fifo 파일을 해당 경로에 생성.**

**3)unlink: 참조하는 count가 0 인경우 실제 파일을 삭제 진행하는 게 좋음 – 삭제 시 unlink 사용**

**4)read/write: 파이프에 정보를 읽고 쓴다.**

**5)open: 특정 이름의 파이프를 열 때 사용**

**3.프로그램 구조**

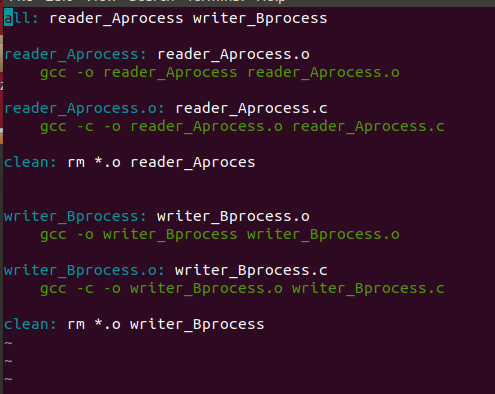
**1)reader: 해당 파일의 count값을 받아서 unlink가 필요한 경우 먼저 진행해준다. 이 후 fifo 파일을 생성하고 열어준다. 연결된 파이프에서 반복적으로 값을 읽어온다.**

**2)writer: 파이프를 열고 반복적으로 사용자에게 입력을 받아 메시지로 만든 후 전송한다.**

1. **IPC 메커니즘 설명 – 이론과 연관**

**IPC에는 메모리를 공유하는 방식과 메시지를 전달하는 방식이 있는데, 이 방법은 메시지를 전달하는 방식에 해당한다. 하나의 파이프(fifo 파일이 대신함)에 보내는 쪽과 받는 쪽이 정해져 있고 sender가 메시지를 파이프에 쓰게 되면 차례로 reader가 읽게 된다. 단방향임을 주의해야 한다.**

1. **컴파일 방법 설명**



**all 키워드를 통해서 여러 프로그램을 동시에 컴파일 하였다.**

**c파일로부터 오브젝트 파일을 생성하고 그 오브젝트 파일을 컴파일 하여 실행파일을 만들었다.**

* **Message Queue**

1. **프로그램 설명**

**1.자료구조:**

**-큐의 형태로 FIFO이지만 투입한 데이터에 번호를 붙여 중간 데이터도 접근이 가능**

**-고유한 key값을 갖아 식별이 가능하다.**

**-메시지의 자료구조는 구조체로 사용자가 만들어서 사용하되 long 타입의 msgtype 멤버변수가 반드시 필요**

**2.함수:**

**1)msgget: 메시지 큐를 생성하거나 생성된 객체 참조에 사용한다.**

**2)msgrcv: 메시지 큐에 메시지를 넣을 때 사용한다.**

**3)msgsnd: 메시지 큐에서 메시지를 읽어 올 때 사용한다.**

**3.프로그램 구조**

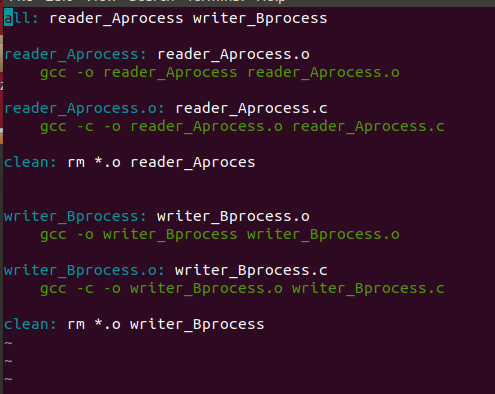
**1)reader: 큐의 식별자를 받아오거나 없다면 생성한 후에 while 문 내부에서 종료조건에 걸리기 전까지 큐에서 메시지를 읽어오고 출력하기를 반복한다.**

**2)writer: 큐의 식별자를 받아오거나 없다면 생성한 후에 while문 내부에서 종료조건에 걸리기 전까지 input을 받아 메시지로 만들고 큐에 넣기를 반복한다.**

1. **IPC 메커니즘 설명 – 이론과 연관**

**IPC에는 메모리를 공유하는 방식과 메시지를 전달하는 방식이 있는데, 메시지 큐는 메시지를 전달하여 통신하는 방식이다. 두 프로세스가 하나의 메시지 큐를 통해서 통신하게 되는데 한쪽에서 msgsnd를 통해 큐에 메시지를 넣어두면 msgrcv로 다른 프로세스가 꺼내서 읽을 수 있게 된다.**

1. **컴파일 방법 설명**



**all 키워드를 통해서 여러 프로그램을 동시에 컴파일 하였다.**

**c파일로부터 오브젝트 파일을 생성하고 그 오브젝트 파일을 컴파일 하여 실행파일을 만들었다.**

* **Shared Memory**

1. **프로그램 설명**

**1.자료구조**

**1)공유 메모리: 여러 프로세스가 공동으로 사용하는 메모리를 의미하며 physical 메모리에 해당**

**2.함수**

**1)shmget: 키 값을 통해 공유 메모리의 id를 얻어오는 함수**

**2)shmat: 프로세스의 메모리 공간에 공유 메모리를 붙이는 함수**

**3)shmdt: 프로세스의 메모리 공간에서 공유 메모리를 떼는 함수**

**4)shmctl; 공유 메모리의 정보를 확인/변경/제거하는 함수**

**3.프로그램 구조**

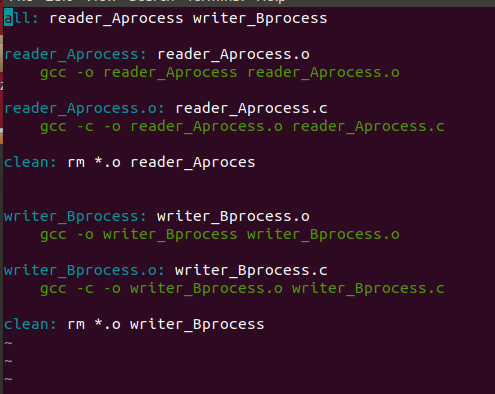
**1)reader: 먼저 특정 키 값을 통해 공유 메모리의 id를 받아온 후 해당 id의 메모리를 프로세스에 붙인다. 이 후 메시지를 반복적으로 읽어오는데, 널 값이라면 그냥 continue하고 그게 아니라면 새 메시지가 들어온 것 임으로 읽어서 출력한 후 읽은 메시지는 지워줄 것 이기 때문에 다시 해당 메모리의 값을 널로 변경한다. 모든 메시지를 읽었다면 공유메모리를 프로세스에서 떼고 메모리 포인터를 삭제한다.**

**2\_writer: 특정 키 값을 통해 공유 메모리의 id를 받아온 후 그 메모리 공간을 프로세스에 붙인다. 이 후 반복적으로 사용자에게 input을 받고 그 메모리에 쓴다. 모든 입력이 종료되었다면 공유 메모리를 뗀다.**

1. **IPC 메커니즘 설명 – 이론관 연관**

**IPC에는 메모리를 공유하는 방식과 메시지를 전달하는 방식이 있는데, shared memory 방식은 메모리를 공유하여 통신하는 방법이다. 동일한 메모리를 붙이고 있는 프로세스끼리 읽고 쓰기를 통한 통신이 가능해진다.**

1. **컴파일 방법 설명**



**all 키워드를 통해서 여러 프로그램을 동시에 컴파일 하였다.**

**c파일로부터 오브젝트 파일을 생성하고 그 오브젝트 파일을 컴파일 하여 실행파일을 만들었다.**

1. **과제 B**
2. **멀티 스레딩 내용**
   1. **프로세스**
      1. **프로세서에 의해 수행되는 프로그램의 단위**
      2. **독자적 자원을 보유함 -> 자식 프로세스는 메모리는 공유하지 않고 복사만 한다.**
   2. **스레드**
      1. **제어 흐름으로 프로세스 실행 부분으로 실행의 기본 단위**
      2. **실행을 위해 최소한의 자원만을 추가하고 서로 다른 스레드와 대다수의 자원 공유**
      3. **프로세스와 스레드는 어떤 것을 공유 자원을 쓰는지에 차이가 있다.**
         1. **자신만의 주소 공간을 갖고 있다면 프로세스. 그렇지 않다면 스레드이다**
   3. **멀티 스레드**
      1. **멀티 프로세스의 경우 fork를 통해 실현하지만 이 경우 pthread 사용**
      2. **장점**
         1. **context switching이 아니므로 빠르기 때문에 부담이 적다**
      3. **단점**
         1. **하나의 에러가 날 경우 다른 스레드도 영향을 받는다**
         2. **하나의 자원을 동시에 같이 쓰는 과정에서 문제가 발생한다 -> lock이 필요해서 또다른 overhead 의 가능성**