인덕대학교

입 · 출력시스템

– 입 · 출력장치의 통신 방식

컴퓨터소프트웨어학과

김병국 교수

학습목표

- 프로세서와 입 · 출력장치 간의 통신 방식을 안다.
- 프로세서내 입 · 출력모듈을 포함한 전반적인 구조를 안다.
- 입 · 출력장치와 데이터 송/수신을 위한 방식들을 안다.
- 폴링과 인터럽트를 실습을 통해 익힌다.

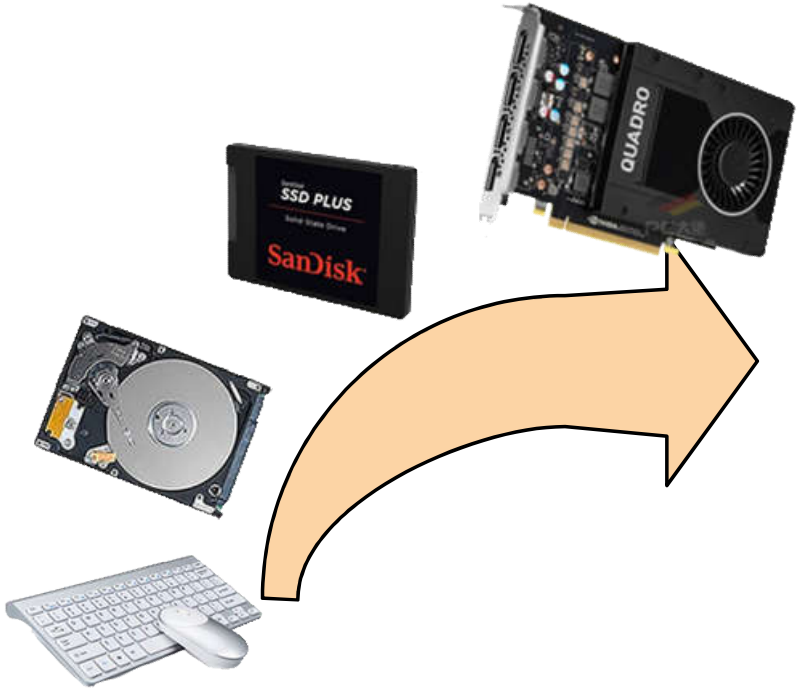
목차

1. 입출력장치와 통신 (1/4)

인덕대학교

❑ 데이터 전송 속도에 따른 분류

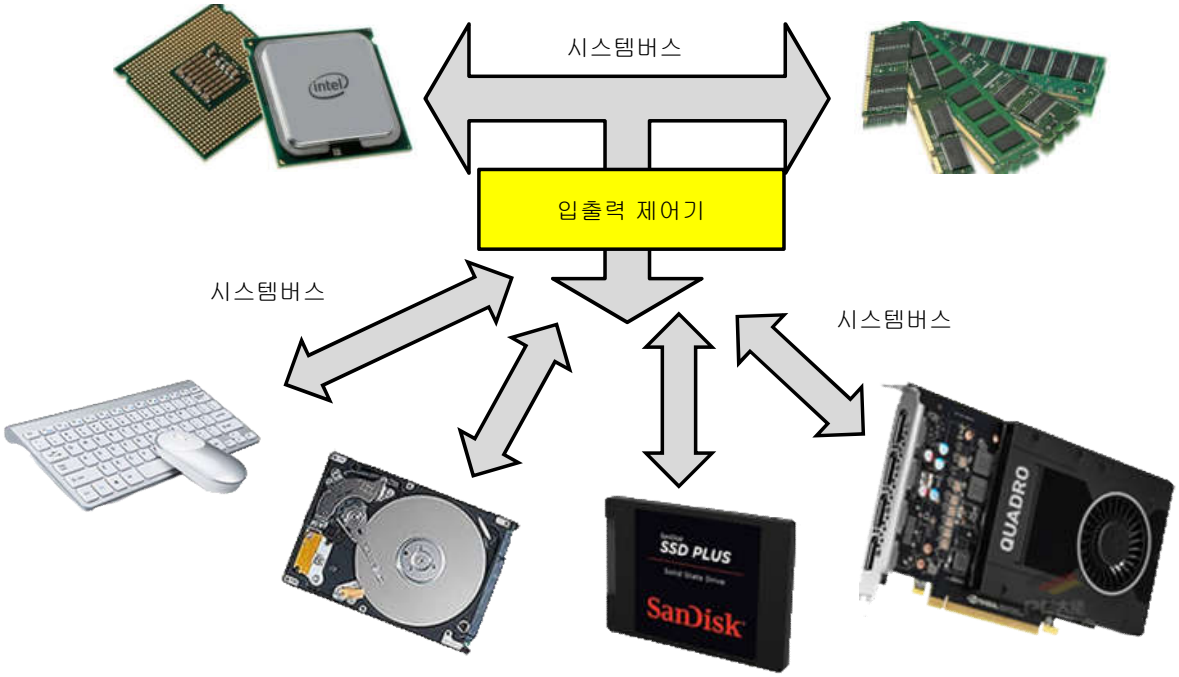
- 저속의 주변장치(키보드, 마우스 등)와 고속 주변장치(그래픽 카드, 하드 디스크 등)로 나뉨
- 하나의 버스로 주변장치를 묶으면 저속 장치로 인해 고속 장치의 활용성이 떨어짐 → 효율성 저하 발생



1. 입출력장치와 통신 (2/4)

❑ 버스의 기본적인 연결 구조

- CPU와 주기억장치(RAM)를 주 축으로 다른 입출력장치와 통신함
- 모든 통신은 시스템 버스(FSB)를 통해 가능
  - 주소, 제어, 데이터 버스로 구성



1. 입출력장치와 통신 (3/4)

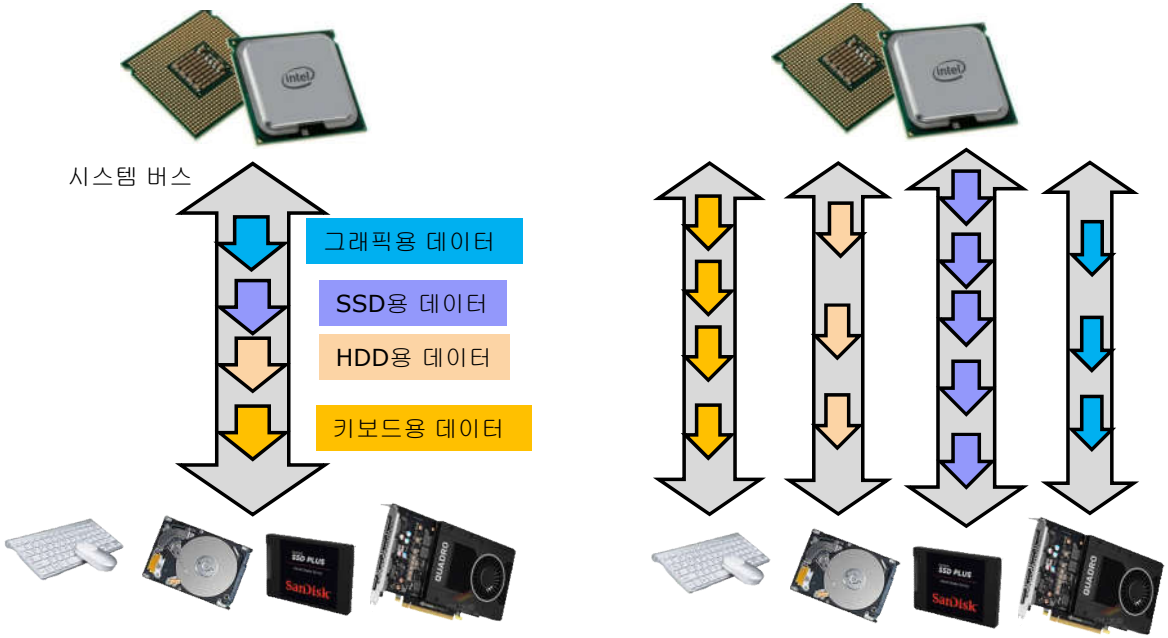
인덕대학교

❑ 채널(Chanel)

- 데이터가 지나가는 통로
- 버스(BUS)를 통한 논리적인 경로

❑ 채널 공유와 분리 (1/2)

- 서로 다른 전송 대역(bandwidth)를 갖는 장치들 간 채널을 공유 또는 분리



❑ 채널 공유와 분리 (2/2)

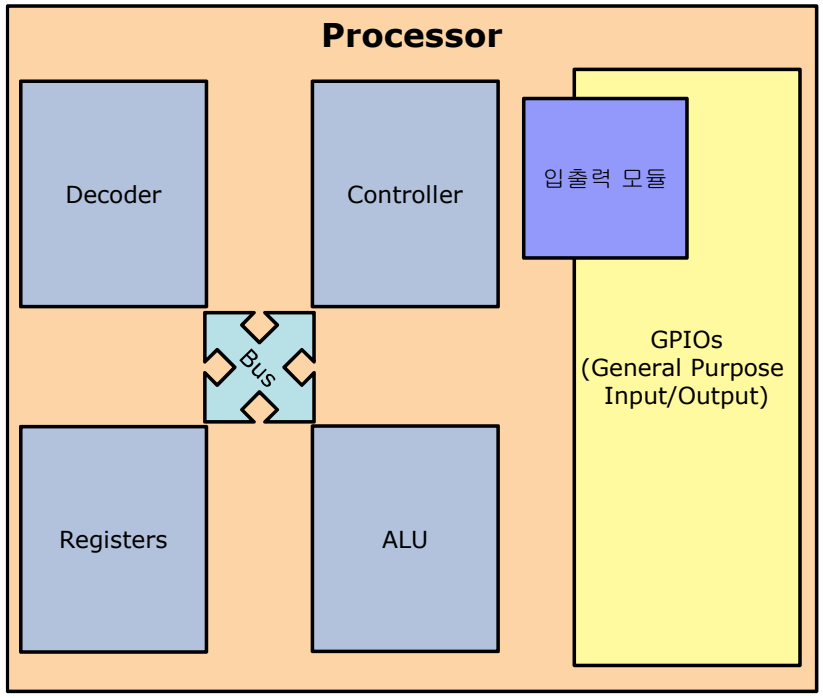
- 공유
  - 공유되는 통로를 통해 모든 연결된 주변 장치들과 통신을 수행
  - 채널의 구성이 단순 → 저렴
  - 모든 주변 장치의 성능은 가장 낮은 성능의 주변장치의 성능을 따름 → 비효율적

- 분리
  - 각 주변장치를 위한 별도의 채널을 구성
  - 채널의 구성이 복잡 → 고가
  - 성능향상 기대

2. 입출력 모듈의 구성 (1/2)

❑ 일반적인 구성

- 프로세서마다 입출력장치를 위한 내부 모듈(유닛)들이 각기 다른 형태로 구성
- 많은 프로세서들이 GPIO를 통해 외부 장치(주기억장치 포함)와 연결됨
  - 외부 버스(FSB) 또한 GPIO의 일종



2. 입출력 모듈의 구성 (2/2)

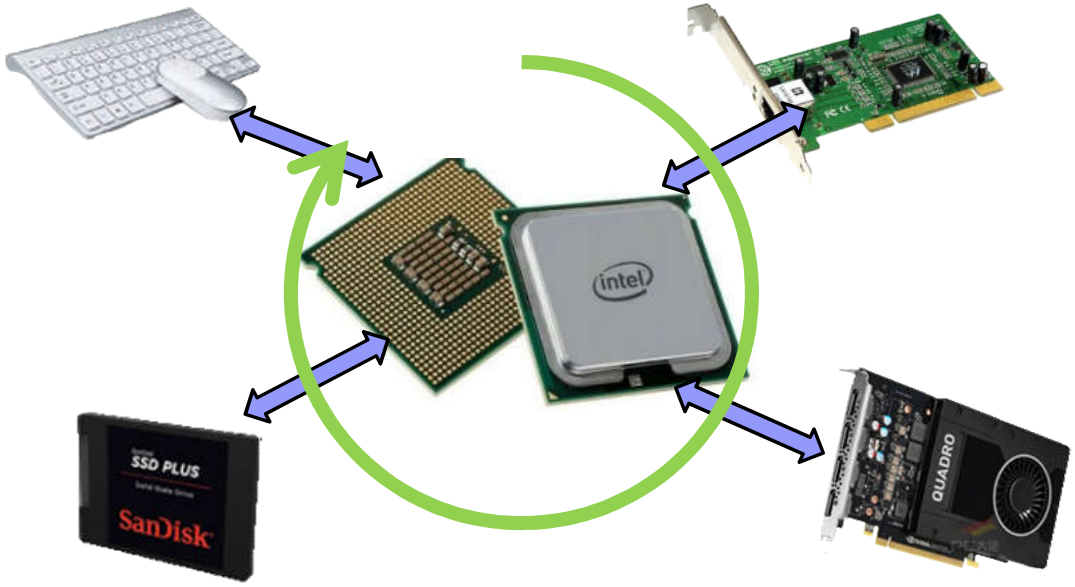
❑입출력 모듈 기능

- 내부 자원과 데이터 입출력 등 다양한 동작을 제어
- 입출력장치에 명령을 보내기 위해서는 장치의 식별을 위한 주소 등을 지정
- 타이밍 동기 기능을 제공
- 버퍼링을 이용한 전송속도 조절
- 오류 검출

3. 데이터 처리 방식 (1/7)

❑ 폴링(Polling)

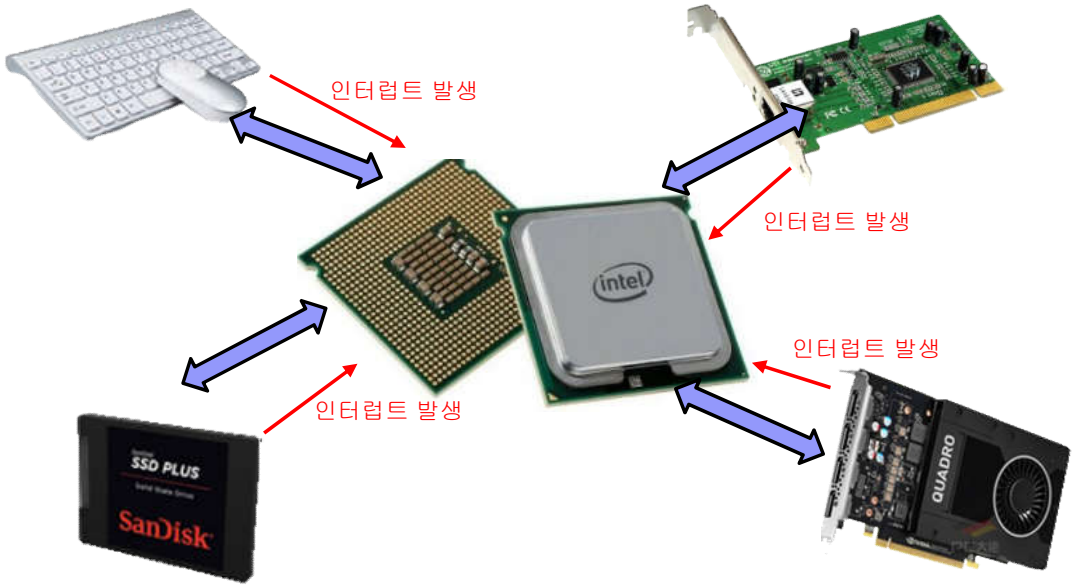
- CPU가 연결된 모든 주변장치에게 데이터 송수신을 위해 상태를 물음
- 동일 작업을 주기적으로 수행
- 잦은 데이터 송수신이 있는 주변장치의 접근에 유리
- 이벤트성의 데이터송수신 장치의 경우 비효율적인 접근이 됨
- CPU의 부하를 발생



3. 데이터 처리 방식 (2/7)

❑ 인터럽트(Interrupt) (1/5)

- 주변장치에서 송수신관련 데이터가 있으면 CPU에게 알림
- CPU는 알림신호를 받으면 해당 주변장치와 송수신작업을 수행
- CPU의 부하를 최소화
- 너무 잦은 인터럽트는 CPU의 성능을 저하시킴

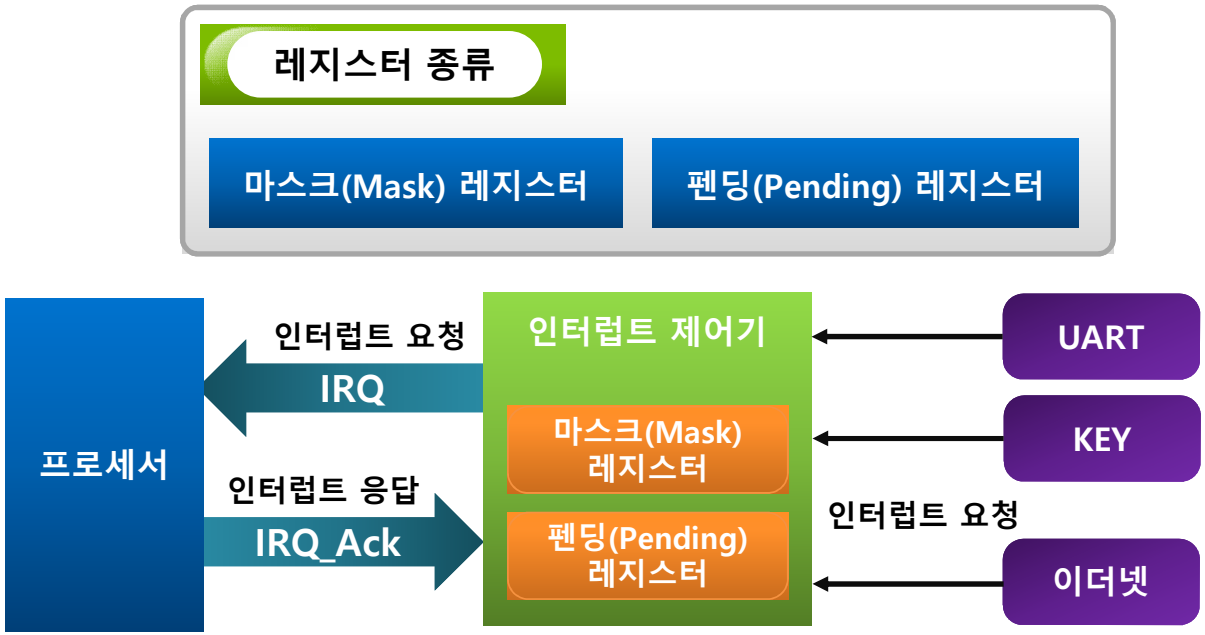


❑ 인터럽트(Interrupt) (2/5)

- 프로세서 코어(core)는 한번에 한 개의 명령만을 수행
- 인터럽트를 이용하면 멀티태스킹을 지원할 수 있도록 함
  - 사용자는 모든 작업이 동시에 수행되는 것 처럼 보임

❑ 인터럽트(Interrupt) (3/5)

- 인터럽트 제어기
  - 입출력 장치에서 발생되는 인터럽트의 요청을 제어함
  - 하드웨어에 따라 인터럽트 응답을 위한 신호도 제공함



3. 데이터 처리 방식 (5/7)

❑ 인터럽트(Interrupt) (4/5)

- 인터럽트 요청과 서비스 실행

인터럽트 요청

- Interrupt Request
- 외부장치에서 입출력 동작에 대한 처리를 프로세서에 요청함

인터럽트 서비스 실행

- 하드웨어적으로 이루어짐
- 일련의 동작(ISR: Interrupt service Routine)을 수행

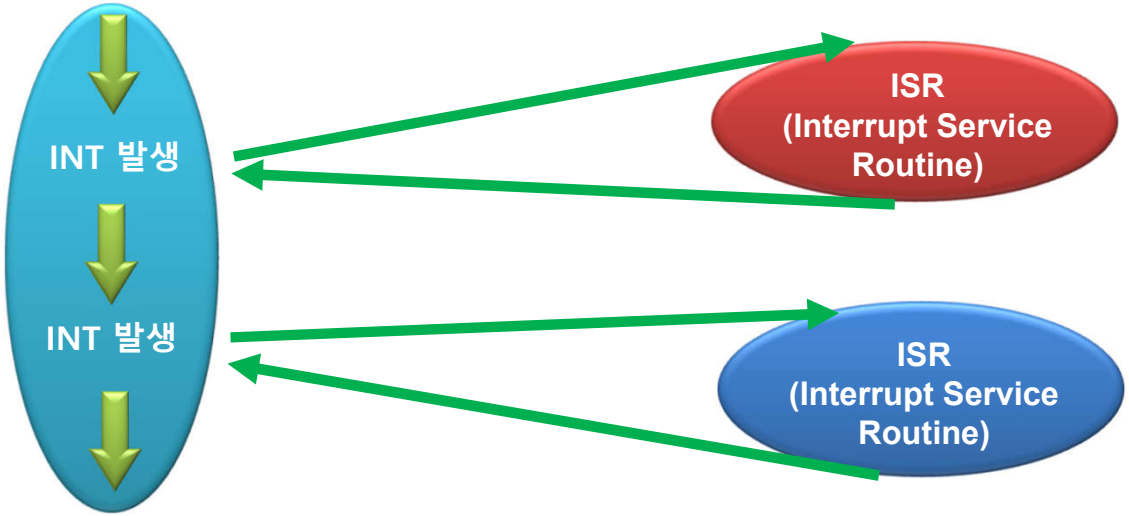
인터럽트 백터(Vector)

- 인터럽트 서비스 루틴을 처리하기 위한 명령 또는 위치가 저장된 메모리 공간

3. 데이터 처리 방식 (6/7)

❑ 인터럽트(Interrupt) (5/5)

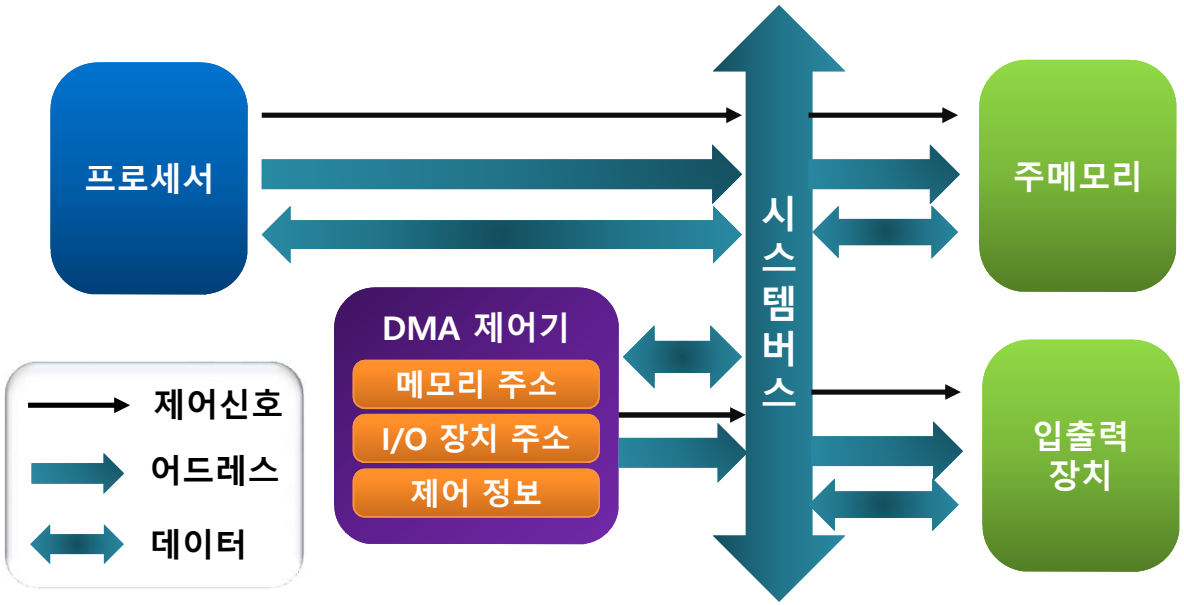
- 인터럽트 처리



3. 데이터 처리 방식 (7/7)

❑DMA(Direct Memory Access)

- 프로세서의 개입 없이 입출력장치가 주기억장치에 직접 접근하는 방식
- 다른 장치 간에도 메모리의 접근을 통해 서로 데이터를 교환할 수 있음



4. 프로그래밍 실습 (1/2)

❑ 폴링 예

- 파일 중에 test.txt 파일이 존재하면, 문자열을 출력하는 프로그램 작성

```c
#include <unistd.h>
#include <stdio.h>

int
main(void)
{
    for(int i=0;;i++)
    {
        if( access("test.txt", F_OK) == 0)
        {
            printf("There is a file(%05d).\n", i);
        }

        usleep(5000);
    }

    return 0;
}
```

4. 프로그래밍 실습 (2/2)

❑ 인터럽트 동작 예

- 10 또는 12번 시그널을 입력 받으면, 문자열을 출력하는 프로그램 작성

```c
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>

void callback_sig_handler(int nSigNumber);

int
main(void)
{
    printf("PID: %d\n", getpid());

    signal(SIGUSR1, callback_sig_handler);
    signal(SIGUSR2, callback_sig_handler);

    for(int i=0; ;i++)
    {
        if(i%4==0)
            printf("\r-");
        else if(i%4==1)
            printf("\r\\");
        else if(i % 4 == 2)
            printf("\r|");
        else
            printf("\r/");

        fflush(stdout);
        usleep(50000);
    }

    return 0;
}

void callback_sig_handler(int nSigNumber)
{
    printf("We just got a signal(%d).\n", nSigNumber);
}
```

수고하셨습니다.