Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 hyungjun Yu(유형준)
love592946@nave.rcom

0405 숙제

모르는거 알아보기/ 책 숙제

socket 함수를 공부하다 보면 PF_INET 이런게 나온다. 이게 ipv4 인터넷 프로토콜을 사용한다.
그럼 ipv4 이건 무엇인지 알아보자

IPV4

네트워크 계층에서 사용하는 ip 주소.
IP 주소의 길이는 4Byte(32Bit)이다.
전세계에 있는 IP 주소의 갯수는 2 의 32 개승이다.(약 42 억개)

전세계 사람들이 스마트폰 등 많이 쓰니까 ip 가 부족해서 나온것이 Ipv6 가 나옴.

SOCKET()함수

헤더
#include<sys/types.h>
#include<sys/socket.h>

형태
int socket(int domain, int type, int protocol);

인수
int domain : 인터넷을 통해 통신할 지, 같은 시스템 내에서 프로세스 끼리 통신할 지의 여부를 설정.

-PF_INET, AF_INET : Ipv4 인터넷 프로토콜을 사용.
-PF_INET6 : Ipv6 인터넷 프로토콜을 사용.
-PF_LOCAL, AF_UNIX : 같은 시스템 내에서 프로세스 끼리 통신.
-PF_PACKET : Low level socket 을 인터페이스를 이용.
-PF_IPX : IPX 노벨 프로토콜을 사용.

Int type : 데이터의 전송 형태를 지정하며 아래와 같은 값을 사용할 수 있음.

-SOCK_STREAM : TCP/IP 프로토콜을 이용.
-SOCK_DGRAM : UDP/IP 프로토콜을 이용.

비유 1. 회사

fork() 와 exec() 함수.

fork()는 현재 실행되는 프로세스에 대해 복사본 프로세스를 생성
생성한 프로세스는 부모프로세스, 새로 생긴 프로세스는 자식 프로세스.

모든 프로세스는 (참고: 최상위 프로세스인 `init` 는 pid 1 을 가진다) 생성될 때 프로세스 아이디를 부여받는다. `fork()` 함수를 이용해서 자식 프로세스를 생성할 수 있는데, 이 때 반환값은 부모에겐 자식 프로세스의 pid, 자식에게는 0 이 된다. 이를 이용하여 프로세스별로 실행할 작업을 만들어줄 수 있다.

예제))

```
int main(void)
{
pid_t pid; //프로세스 번호 (Process ID; PID)가 저장될 변수 pid
pid = fork(); //자식 프로세스 생성.
//성공시 0, 실패시 -1, 부모 프로세스면 양수 (프로세스 ID)를 반환.

If (pid <0) {    //pid 가 음수면 에러난것.
fprintf(stderr, “fork failed”);
return -1;
}
else if (pid == 0)  // pid 가 0 이면 자식 프로세스.
{
printf(“자식 프로세스 (#%i)”, getpid());
}
else{
//pid 가 양수면 부모프로세스
```

printf(“부모 프로세스 (#%i)”,getpid());

wait(NULL); //자식 프로세스들이 모두 종료될때까지 기다림

}

return 0;

}

1.자식프로세스 내(pid == 0 구간)에서 또 fork()를 사용하여 손자 프로세스까지 만들 수 있음.

2.부모 프로세스에서 wait(NULL) 함수가 불러지면 자식 프로세스가 끝날 때까지 기다린 후 진행,

3.주의: pid 는 변수명이지 실제 pid 를 의미하지않음. 따라서 getpid()를 사용해야 이 함수를 부른 프로세스의 id 를 얻을수있다.

Exec()

exec 는 어떤 일을 하는 family 명칭으로 보는 게 더 정확.
Exec family 가 하는 일은 현재 실행되는 프로세스에서 다른 프로세스 일을 하게 하는 것.
즉, 현재 실행중인 프로세스의 메모리를 다른 프로세스를 위해 사용.
예를 들어 현재 실행중인 프로세스를 종료하고 “ls test.txt”명령을 수행하도록 하는 코드.

Execl(“/bin/ls”,”ls”,”test.txt”,NULL)

여기서 첫번째 인자는 실행시킬 함수의 경로, 두번째 인자는 실행시킬 함수명 , 세번째 인자는 함수 실행시 전달할 인자 , 네번째는 환경변수를 전달하는 데 사용.
exec 계열의 함수들을 소개 해보면
int execl(const char *path, const char *arg, …);

```
int execv(const char *path, const *const argv[]);
int execle(                    ,const char arg, … , char *const envp[]);
int execve(                    ,char *const argv[], char *const envp[]);
int execlp( const char *file , const char *arg, …);
int execvp(const char *file, char *const argv[]);
```

exec 뒤의 문자 l , v , p , e 의 의미는 다음과 같다.

L : list 형태의 인자를 취함
v : vector 형태의 인자를 취함
p : 명령을 PATH 에서 검색하겠다
e : 환경 변수 인자를 받는다.

비유 2.
FIFO 와 context switch

FIFO 는 검색해서 봐도 잘 안나와있고 모르겠으므로 쓸게 없음 ㅜㅜ

ㄱ.switch 조건문

```
switch ( 조건 수식 ){

 case 상수값 1:
 명령문 1;
break;

 case 상수값 2:
 명령문 2;
 break;

 default:
 명령문 3;
 }
```

switch 문은 "조건 수식"을 연산한 결과 값을 이용하여 case 별로 명시한 상수값들과 같은지
비교하고, 결과 값과일치하는 case 에 명시된 명령문을 수행하게 됩니다. 예를들어, "조건 수식"의
연산결과가 "상수값 1"과 일치한다면 "명령문 1"을 수행하게 됩니다.

switch 문은 비교시에 위쪽 case 부터 아래쪽 case 순으로 상수값을 비교하며 "조건 수식"의 연산값이
case 키워드에 명시된 각 상수값과 일치하지 않는 경우, 가장 아래쪽에 명시된 default 키워드에 있는
"명령문 3"을 수행하게 됩니다.

default 키워드는 if 문에서 배운 else 문과 유사한 역할을 하고 필요없는 경우에는 생략해도 됩니다.

비유 3.
Signal 함수

어떤 이벤트(인터럽트, 타이머종료)가 발생 했을 때 운영체제가 프로그램에 이를 알리는 기법

종류	이벤트	기본동작
SIGALRM	알람 타이머의 만료	프로그램 종료
SIGCHLD	자식 프로세스가 종료됨	시그널 무시
SIGINT	인터럽트 문자 (Ctrl C)입력됨	프로그램 종료
SIGIO	소켓에 대해 I/O 가 준비됨	시그널 무시
SIGPIPE	종료된 소켓에 쓰기를 시도할 때	프로그램 종료