TI DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)
Gccompil3r@gmail.com
학생-윤지완
Yoonjw7894@naver.com

APPEND:파일이 계속해서 덮어쓰는 작업을 한다.(파일 포인터 맨 마지막에 간다.)
strncpy:같거나 작거나 할 때 → “0”
atoi=문자열을 정수형태로 변환해주는 함수.

Test,90
name[0]=t
name[1]=e
name[2]= s
name[3]= t

strncpy(buf, name, strlen(name));
배열 name 에 있는 값의 길이만큼 buf 에 저장되고 이것을 복사한다.
하지만 이렇게 복사를 하면 이름뒤에 쓰레기 값이 복사가 되기때문에 이것을 보완할 다음 명령어가 필요하다. 그래서 다음명령어를 쓴다.
cur_len = strlen(buf);
버퍼 안에 저장되있는 name 의 문장 길이 만큼만 cu_len 에 저장이 되는 것이다.

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


int main(void)


{
int fd;
fd=open("a.txt",O_WRONLY|O_CREAT|O_TRUC,0644);
close(1);
close(fd);
dup(fd);//복사기능
printf("출력될까?\n");
```

```
return 0;
}
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>


int main(void)


{
int fd;
char buf[1024];
fd=open("a.txt",O_RDONLY);
close(0);
gets(buf);
dup(fd);
printf("출력될까?\n");
return 0;
}
```

gets=입력을 받는 함수.

gets 는 fd 에서 받은 입력을 받아 dup 가 받고 dup 를 출력시키면 fd 에 값이 출력 되고 close(0)이므로 파일에서 입력을 받겠다는 것이다.dup 는 최근에 close 된 데이터만 받는다.

Ps -ef=현재 사용되고 있는 프로세스 꺼지지않는 프로세스를 보여준다.

Ps -ef|grep bash= grep 으로 bash 를 찾고 있는다.

Ps -ef|grep bash|grep -v grep=bash 를 제외한 나머지 를 grep 로 찾는것

tail -c 6 mycat.txt:문자수

tail -n 6 /var/log/messages:라인수

lseek=함수는 파일포인터의 위치를 특정 지점을 기준으로 얼마만큼 움직여 주는 함수입니다.

```
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>
```

```
int main(int argc, char *argv[])


{
int fd;
char ch = 'a';
fd=open(argv[1],O_WRONLY|O_CREAT|O_TRUNC,0644 );
lseek(fd,512 - 1,SEEK_SET);
write(fd, &ch,1);
close(fd);
return 0;
}
```

512byte 부분에 매직 넘버를 붙인다.그 이유는 이 동작이 끝나고 다음동작을 언제 시작하는지 쉽게 알아볼 수 있어서 이다.

cpu 는 프포세스를 관리한다.
프로세스는 cpu 의 추상화다.
메모리(DRAM)의 정보들이 분활되어 쌓여있다.그 안을 까고 보면 기계어가 있다.기계어는 범용 레지스터를 제어한다.
PID= 프로세스 고유번호
CPU 안에 범용 레지스터들이 존재한다.

1.프로세스가 여러개 있는데 어떻게 동시에 실행되는가?
1)정말 동시에 가능한가?->싱글코어일때
2)병렬 처리는 뭐지?->멀티 코어
컴퓨터 구조론
:cpu 는 오로지 한 순간에 한 가지의 연산만 수행한다.
cpu 의 주파수는 2GHZ->f
f=1/t,1/f=t(주기)
:주파수가 증가하면 주기가 짧아져서 엄청나게 빠르다.
아주 빠른속도로 여러 프로세스들이 제어권을 넘겨주면서 cpu 를 사용한다면 우리가 느끼지 못하는 순간에 모든 작업이 완료된다.

Contxt switching

A	B
mov ax,7	mov bx,100
add ax,bx	mov ax,200
mult ax,2	add ax,bx

A 라는 프로세스와 B 의 프로세스가 있다.이 둘은 각각 다른 동작을하는데 동작을 하는 부분에서 동작을 하다가 제어권을 넘어가는 경우가 있다. 속도가 너무빨라서 그럼 그때 ax=7 을저장하고 B 로 넘어가면 BX 100 AX 200 이 동작하고 다시 A 프로세스로 와서 AX=200,BX=100 이 되고 이둘을 더해 총 600 이 결과값이 나온다.B 프로세스도 마찬가지이다.그래서 이부분을 해결하고자 처음 AX 에 7 을 넣고 그 값을 CPU 별도으 저장장소에 저장하고 A 프로세스 제어권이 생기면 그값을 불러와 동작을 실행하는 것이다. 이 동작을 COTXT SWITCHING 이다.
CPU 는 프로세스에게 동작을 하게 하는데 일정시간만 주고 그 시간이 지나면 동작이 멈추고 다음 할당된 시간에 다음동작을 하게 된다.

각 프로세스는 cpu 의 점유율을 높이기 위해 강한 경쟁을 하는데 이 부분에서 생겨난 것이 우선순위이다.

Multi-tasking

cpu 에서 우선순위에 의해서 동작을 하는데 이런 우선순위만 따지면 하던 프로세스만 동작을 하기때문에 이것을 보완하고자 대기 시간을 을 체크해서 시간이 너무 길면 그 프로세스부터 실행하며 만약 A 프로세스가 동작하다가 interrupt 의 요청이 들어오면 A 프로세스는 waiting 프로세스에 두고 interrupt 를 동작을 하고 동작하던 A 프로세스를 다시 불러와서 실행하는 방식이다.

```
#include<string.h>
#include<stdio.h>
#include<fcntl.h>

int main(void)

{
int fd,ret;
char buf[1024];
mkfifo("myfifo");
fd=open("myfifo",O_RDWR );
for(;;)
{
ret=read(0,buf,sizeof(buf));
buf[ret -1]=0;
printf("keyboard input :[%s]\n",buf);
read(fd,buf,sizeof(buf));
buf[ret -1]= 0;
printf("pipe input:[%s]\n",buf);
}
return 0;
}
```

컴파일 하고 mkfifo myfifo 를 쓰고 실행하면 터미널 한개를 더 open 해서 cat> myfifo 를 쓰면 두 터미널 사이에 메시지를 주고 받을수 가 있다.

block 과 charactor 와의 차이점

block(b):특정 단위를 가지고 움직인다.(하드 디스크,DRAM)
제어권이 넘어가지 않는다.

Character(c):4KBYTE 가 최소 단위이다.순서가 존재한다.

```
#include<string.h>
#include<stdio.h>
#include<fcntl.h>

int main(void)

{
int fd,ret;
char buf[1024];
```

```
fd=open("myfifo",O_RDWR );
fcntl( 0,F_SETFL,O_NONBLOCK);
fcntl(fd,F_SETFL,O_NONBLOCK);

for(;;)
{
if(( ret= read(0,buf,sizeof(buf)))>0)
{
buf[ret -1]=0;
printf("keyboard input :[%s]\n",buf);
}
if(( ret= read(fd,buf,sizeof(buf)))>0)
{
buf[ret-1]= 0;

printf("pipe input :[%s]\n",buf);
}
}
close(fd);
return 0;
}
```