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

```
#include<sys/types.h>
#include<stdio.h>
#include<dirent.h>

int main(void)

{
	DIR *dp;
	 int i=0;
	struct dirent *p;
	 dp= opendir(",");

	while(p=readdir(dp))//디렉토리에 있는 내용을 불러온다
	{
		if((p->d_name[0] == ","))

		continue;
printf("%-16s ",p→d_name);
  if((i+1)%5==0)//5 개의 당위로 갱을해라.
printf("\n");
i++;
		}
printf("\n");
closedir(dp);
return 0;
}
#include<sys/types.h>
#include<dirent.h>
#include<fcntl.h>
#include<unistd.h>

int main(int argc,char **argv)

{

	DIR *dp;
  struct dirent *p;
	 dp= opendir(",");
int i=0,cmd;
cmd = getopt(argc ,argv , "a");
	while(p = readdir(dp))

	{
		if(cmd=="a")
		{

		if(p->d_name[0] == ',')
```

```
continue;
}
}
printf("%-16s",p->d_name);
if((i+1)%5 == 0)
printf("\n");
i++;
}
printf("\n");
closedir(dp);
return 0;
}
```

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


int main(int argc ,char **argv)

{
   DIR *dp;
 int i=0;,cmd;
struct dirent *p;
int flag = 0;
while((cmd=getopt(argc,argv , "a|RI"))>0)
{
switch(cmd)
{
case 'a':
flag |=1;
break;
case 'b':
flag |=2;

break;

case 'R':
flag |=4;
break;
 case 'i':
```

```
flag |=8;
break;
 }

}
```

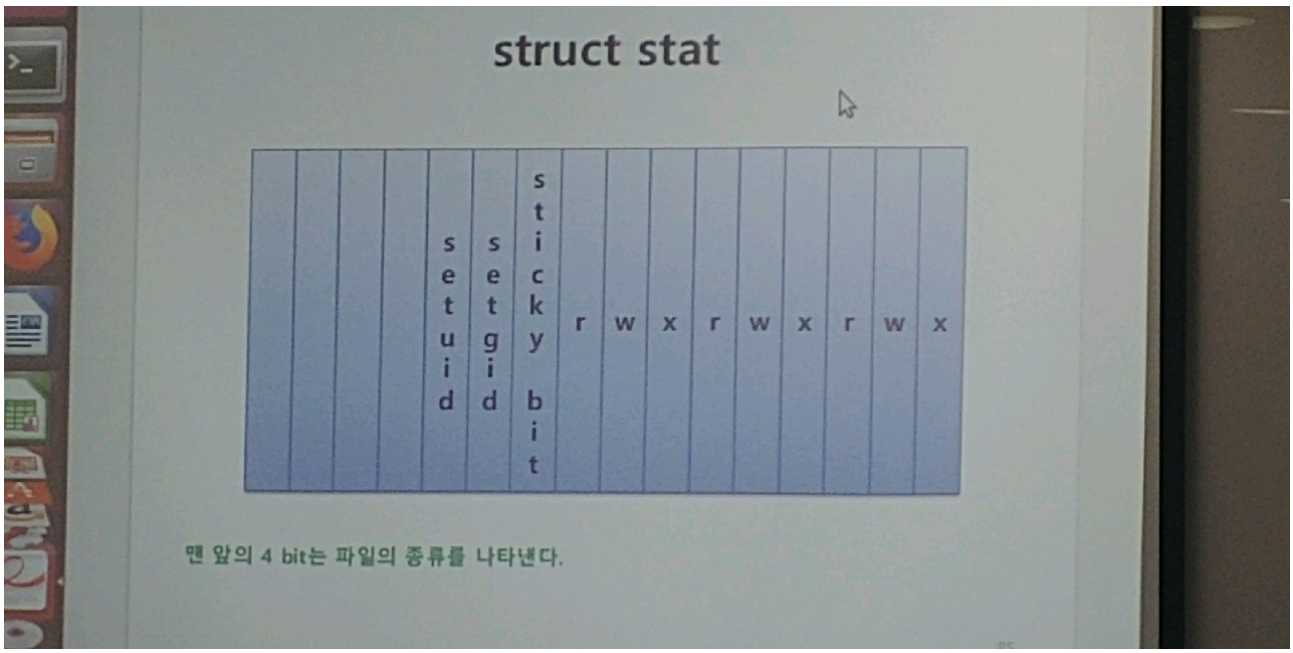
ls -R 모든 디렉토리를 순회한다.

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



int main(int argc ,char **argv)

{
struct stat buf;
char ch;
stat(argv[1],&buf);

if(S_ISDIR(buf.st_mode))
ch='d';

if(S_ISREG(buf.st_mode))
ch='-';
if(S_ISFIFO(buf.st_mode))//파이프
ch='P';
if(S_ISLNK(buf.st_mode))//바로가기
ch='I';
if(S_ISSOCK(buf.st_mode))//소켓
ch='s';

if(S_ISCHR(buf.st_mode))//charactor device
ch='c';
if(S_ISBLK(buf.st_mode))//block device
ch='b';
printf("%c\n",ch);
return 0;
}
```

755:사용자는 실행까지 가능하며 나외 다른사람에세는 쓰기권한을 주지 않는다.
644:사용자는 쓰기권한이 있으며 그 외에는 읽기만 가능하다.

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

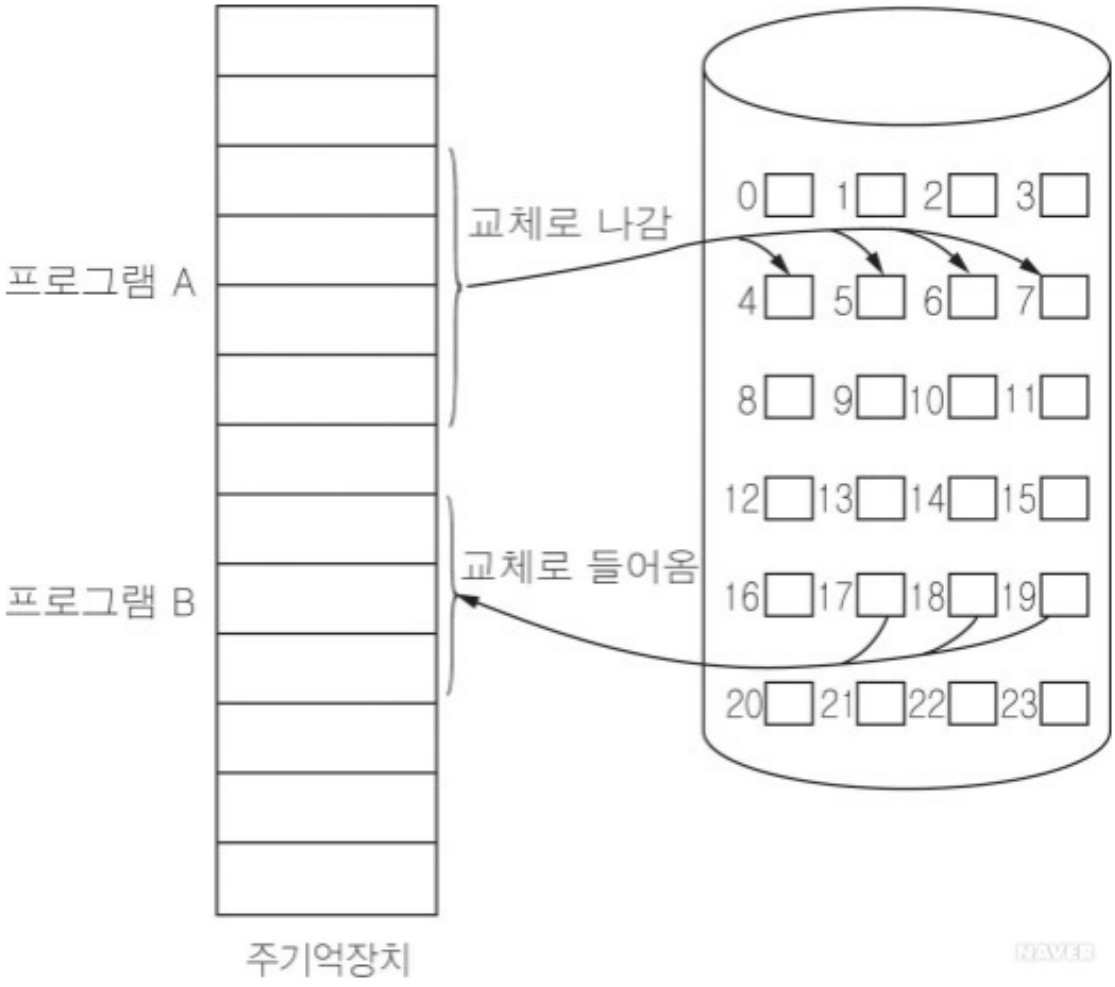
```
{
struct stat buf;
char ch;
char perm[11] = "__________";
char rwx[4] = "rwx";
int i;

stat(argv[1],&buf);


if(S_ISDIR(buf.st_mode))
perm[0]='d';

if(S_ISREG(buf.st_mode))
perm[0]='-';
if(S_ISFIFO(buf.st_mode))//파이프
 perm[0]='P';
if(S_ISLNK(buf.st_mode))//바로가기
perm[0]='I';
if(S_ISSOCK(buf.st_mode))//소켓
perm[0]='s';
if(S_ISCHR(buf.st_mode))//charactor device
perm[0]='c';
if(S_ISBLK(buf.st_mode))//block device
perm[0]='b';

for(i=0;i<9;i++)
if((buf.st_mode>>(8-i))&1)
perm [i+1] = rwx[i%3];
printf("%s\n",perm);
return 0;
}
```

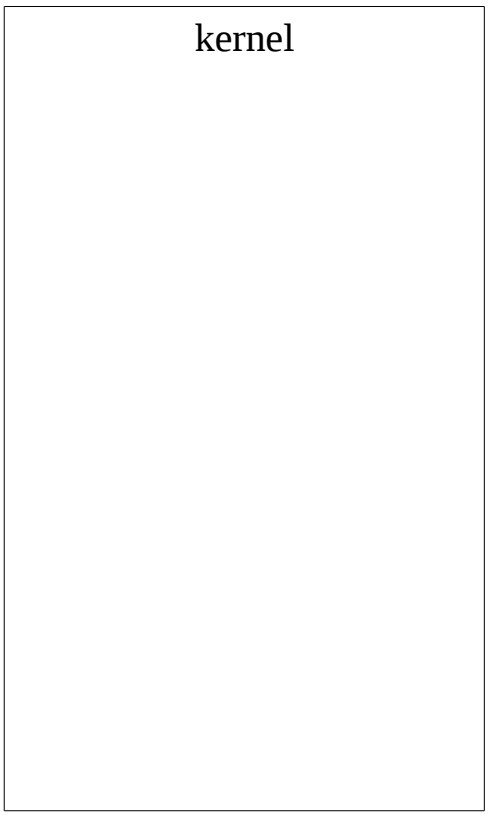


struct stat buf 의 내부 모습은 이렇다.처음 4 비트는 구조체의 상태를 나타내는 부분이고 나머지는 미리 세팅되어있는 상태이다 8 비트 부터 rwx 가 반복되어서 입력되었는데 char rwx[4] = "rwx"; 이 명령어 를 통해서 rwx 를 차례대로 출력시키는 것이다. 이 동작을 하는것은 마지막 부분에서 실행이 된다.처음 perm[1]지에는 rwx 의 0 번지의 문자를 넣겠다는 것이다.

페이징

관련이미지


교체로 나감
프로그램 A
교체로 들어옴
프로그램 B
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
주기억장치



<vm(가상 메모리)>

가상 메모리에서 특정 주소(ex 0xbf884c8c)를 확인하여 주소를 10bit 단위로 나눈다. 이 작업을 한는 이유는 저 큰공간을 한번에 DRAM(물리 메모리)에넣을 수도 없고 넣으면 공간 효율이 너무 않좋기 때문에 주소를 분활을 하여 pgd 동작을 한다. pgd 동작은 이것은 가상 메모리도 아니다.pgd 는 1024 의 크기를 가지고 있으며 처음 10bit 의 766 의 번지를 찾아서 132 의 번지를 찾고 132 번지는 DRAM 의 기본 단위 4KB 의 3212 번지에 넣는다.그럼이 부분에서 STIKY BIT 가 물리 메모리의 최소단위를 ㅍ에 거사ㅓ 디스크에 옮겨주는데 이때 SWAP 이 SUPPORT 해준다. 이 작업을 하는이유는 이 작업을 않하면 프로세싱하는 속도는 빠르지만 사용자는 속도가 느리게 느껴지기 때문에 한 부분을 할당해 다른 작업을 하게 해준다. 이때 디스크에 넘겨졌던 데이터들은 디스크에 저장되어있다가 DRAM 에서 사용을 원할때 다시 들어와서 사용 할 수 있다.