

**Xilinx Zynq FPGA, TI DSP, MCU 기반의
프로그래밍 및 회로 설계 전문가 과정
#21**

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1.배운내용 복습.

수업내용 복습

---sizeof vs strlen---

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

int main(void)
{
    int i,fd;
    int sread;
    char buf[10]="Hellow Hi";
    char buf2[32];
    fd=open("b.txt",O_RDONLY);
    sread=read(fd,buf2,sizeof(buf2));

    printf("sizeof %ld\n",sizeof(buf));
    printf("strlen %ld\n",strlen(buf));
    printf("read %d\n",sread);
    printf("strlen \n");
    write(1,buf2,strlen(buf2));
    printf("sizeof \n");
    buf2[sread -1]='0';
    write(1,buf2,sizeof(buf2));

    printf("printf \n");
    for(i=0;i<10;i++)
```

```
{
    printf("%c",buf[i]);
}

printf("\n");

return 0;
}
```

이 두 개가 자꾸 헷갈려서 소스코드를 짜서 확인해보았다.

buf 와 buf2에는 똑같은 문자열이 저장되어있다.

b.txt 내용도 Hellow Hi이다.

하지만 strlen과 sizeof 의 값을 확인했을때는 서로 값이 달랐다.

strlen은 9 sizeof 는 10 이라는 결과가 나왔다.

sizeof를 했을 때 출력하는 방법이 아직도 헷갈린다.

---ls 구현---

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

int main(void)
{
    DIR *dp;//파일포인터랑 비슷
    int i =0;
    struct dirent *p;
    dp = opendir(".");
    //디렉토리 내에 리스트 현재 디렉토리 나타내는 심볼.
    while(p= readdir(dp))
    //현재 디렉토리를 읽어서 그 포인터를 얻음 p에 디렉토리에 있는 내용들의 리스트
    //들이 넘어옴 값이 있으면 while을돌아
    {
        if(p->d_name[0]!='.')
        //배열의 0 맨앞의 글자가 . 이면 통과 .이 아니면 이름을 출력
```

```

        continue;
        printf("%-16s", p->d_name);
        if((i+1) %5 ==0)
//5번 넘으면 엔터 해 5개 출력하면 엔터해서 출력 결과가 파일 및 폴더 이름이 5
개씩 한라인에 정렬되서 리스트를 보여준다.
        printf("\n");
        i++;
    }
    printf("\n");
    closedir(dp);
    return 0;
}

```

```

a.out      ls.c
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out
a.out      ls.c

```

확인을위해 dp를 이전폴더로 open을 해보았다.

dp = opendir(".."); 으로 바꾸고 출력하면 다음과 같은 결과를 얻을 수 있었다.

```

siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out
17      21      18      Homework      quiz

```

터미널 창에서 cd .. 과 비교해보면 동일하다.

```

siyun@siyun-CR62-6M:~/my_proj/21$ cd ..
siyun@siyun-CR62-6M:~/my_proj$ ls
17 18 21 Homework quiz

```

if 문에서 name에 첫 글자가 . 이면 밑에 내용을 수행하지 않는다.

그래서 if문을 지우고 수행해본 결과 숨겨진 파일 2개 현재폴더 이전폴더가 나왔다.

---ls 구현(a option)---

```

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

int main(int argc, char **argv)

```

```

{
    DIR *dp;
    int i = 0,cmd;
    struct dirent *p;
    cmd = getopt(argc,argv,"a");
    dp = opendir(".");
    while(p = readdir(dp))
    {
        if(cmd != 'a')
//a가 없으면 .을 제긴다.
        {
            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;
}

```

```

siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -a
.      a.out      ls.c      ..      a.c
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out
a.out      ls.c      a.c
siyun@siyun-CR62-6M:~/my_proj/21$

```

a가 있으면 숨겨진 파일도 같이 나오고 a가 없으면 숨겨진 파일이 나오지 않는다.

---ls 구현(state)---

```

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

```

```

int main(int argc, char **argv)
{
    struct stat buf;
//stat상태를 본다 입력한 어떤 정보의 state 정확히 말하면 파일 그정보를 버퍼에
저장 man -s2 stat , man -s3 stat
    char ch; //현재 입력한 파일 상태를 출력하기위한 변수
    stat(argv[1],&buf);
//argv에 입력한 파일의 상태를 확인하여 버퍼에 저장
    if(S_ISDIR(buf.st_mode))
//디렉토리나 st_mode 버퍼에 들어있다
        ch='d';
    if(S_ISREG(buf.st_mode))//일반이나 파일
        ch='-';
    if(S_ISFIFO(buf.st_mode))//파이프나
        ch='p';
    if(S_ISLNK(buf.st_mode))//바로가기 파일이나
        ch='l';
    if(S_ISSOCK(buf.st_mode))//소켓 즉 네트워크 관련된 장치들 요즘은 씨
나 비로 통합되었음
        ch='s';
    if(S_ISCHR(buf.st_mode))//캐릭터 디바이스 캐릭터나
        ch='c';
    if(S_ISBLK(buf.st_mode))//블록디바이스나
        ch='b';
    printf("%c\n",ch);//마지막 최종적으로 들어온 파일에 대한 정보를 출력한
다.

    return 0;
}

//ls -l 맨앞을 지정하는 코드

```

```

siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out ls.c
-
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out ./a.out
-
siyun@siyun-CR62-6M:~/my_proj/21$ mkdir m
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out m
d
siyun@siyun-CR62-6M:~/my_proj/21$

```

./a.out d를 입력하며 c 가 나온다.

sys/stat.h 헤더안에 다이렉트의 상태를 알려주는 소스코드를 활용하여 작성한 소스코드이다.

---a b option---

```

#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    int cmd;
    while((cmd = getopt(argc,argv, "ab")) > 0)
//argc 란 argv를 받는다 argc 인자번호 argv문자열 옵션이 몇개왔는지 몇개를
분석할지 알기위해 두개 받는거다. 그리고 a , b두개의 옵션을 감지한다는 뜻
    {
        switch(cmd)
        {
            case 'a':
                printf("a option\n");
                break;
            case 'b':

```

```

        printf("b option\n");
        break;

    default:
        printf("unknown option\n");
    }
}
return 0;
}

```

---int logic option---

```

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

#define A      (1 << 0)
#define B      (1 << 1)
#define C      (1 << 2)
#define D      (1 << 3)
#define E      (1 << 4)
#define F      (1 << 5)
#define G      (1 << 6)

void check_flag(int flag)
{
    int i, tmp = flag;

    for(i = 0; i < 7; i++)
    {
        switch(tmp & (1 << i))
        {
            case A:
                printf("A\n");
                break;
            case B:
                printf("B\n");
                break;
            case C:
                printf("C\n");
                break;
            case D:
                printf("D\n");
                break;
            case E:
                printf("E\n");
                break;
            case F:
                printf("F\n");
                break;
            case G:
                printf("G\n");
                break;
        }
    }
}

//7비트 있기 때문에 1가 0 이면 tmp 는 0000001이 되고

```

//i 가 1 이면 0000010이 되 이런식으로 케이스문을 제대로 통과한다.
 //flag 값은 메인에서 1<<i값이랑 동일한 값이 들어온다.
 //공책에다 해보니 동일한 값이 들어옵니다..
 //그래서 스위치 문이 통과되어 0000001 일 때 케이스 1을 통과하고
 //0000010일 때 케이스 2를 통과하는 식으로 소스코드가 구성되어있다.

```

{
    case 1:
        printf("A\n");
        break;
    case 2:
        printf("B\n");
        break;
    case 4:
        printf("C\n");
        break;
    case 8:
        printf("D\n");
        break;
    case 16:
        printf("E\n");
        break;
    case 32:
        printf("F\n");
        break;
    case 64:
        printf("G\n");
        break;
}
}
}

```

```

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

```

```

int cmd;
int flag;

/* 7 개의 옵션: a, b, c, d, e, f, g */
while((cmd = getopt(argc, argv, "abcdefg")) > 0)
{
    switch(cmd)
    {
        case 'a':
            flag |= 1 << 0;
            printf("a option\n");
            break;
        case 'b':
            flag |= 1 << 1;
            printf("b option\n");
            break;
        case 'c':
            flag |= 1 << 2;
            printf("c option\n");
            break;
        case 'd':
            flag |= 1 << 3;
            printf("d option\n");
            break;
        case 'e':
            flag |= 1 << 4;
            printf("e option\n");
            break;
        case 'f':
            flag |= 1 << 5;
            printf("f option\n");
            break;
        case 'g':

```

```

            flag |= 1 << 6;
            printf("g option\n");
            break;
        default:
            printf("unknown option\n");
    }
}

check_flag(flag);

return 0;
}

```

---a일 때 숨겨진 파일 표시---

```

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc , char **argv)
{
    DIR *dp;
    int i = 0,cmd;
    struct dirent *p;
    int flag = 0;
    while((cmd = getopt(argc , argv, "abRi"))>0)
    {
        switch(cmd)
        {

```

```

        case 'a':
            flag |=1;
            break;
        case 'b':
            flag |=2;
            break;
        case 'R':
            flag |=4;
            break;
        case 'i':
            flag |=8;
            break;
    }

    }
    dp=opendir(".");
    while(p=readdir(dp))
    {
        if(!(flag &1))
//플래그 & 1 이 거짓일 때 통과 즉 a빼고는 다 통과한다. 따라서 a를 입력했을
//경우 빼고는 숨겨진 파일을 출력하지 않고 a를 입력했을 경우 if문을 통과하지 못하
//여 숨겨진 파일 까지 모두 출력하게된다.
        {
            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;
    }

```

---int logic option(my option)

```

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

int main(int argc, char **argv)
{
    int cmd;
    cmd = getopt(argc,argv,"abclps");

    if(cmd<<0 | cmd << 1)
    {
        if(cmd == 'a')
        { printf("a option\n");
        }
        else if(cmd == 'b')
        { printf("b option\n");
        }

        else if(cmd == )
        { printf("ab option\n");
        }

    }

    /* if(cmd << 2 && cmd <<4 )
    { printf("cp option\n");
    }
    if(cmd << 3 && cmd << 5)
    {
        printf("ls option\n");
    }
    if( ~(cmd == 'a') | ~(cmd=='b') | ~(cmd =='c') | ~(cmd =='l') | ~(cmd =='p') | ~(cmd=='s'))
    {
        printf("not option\n");
    }
    */

```

```
return 0;
```

```
}
```

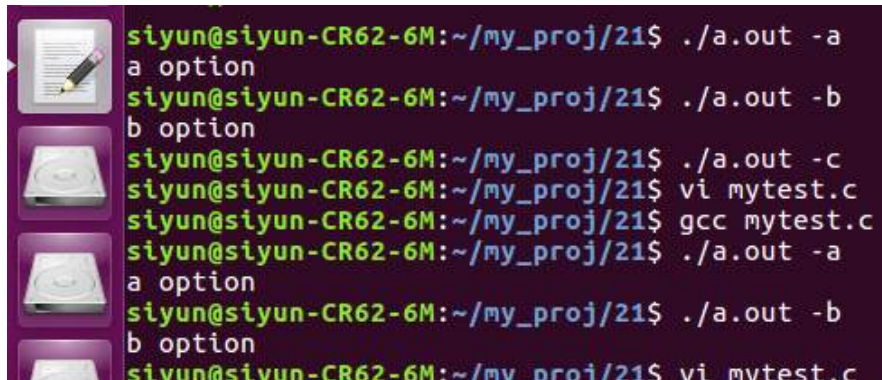
수업시간에 구현하다 말았던 option의 소스코드이다.

or를 해서 a와 b를 하면 a b 옵션이 각각 출력되는건 성공했었지만

내가 원하던 cp, ls option은 성공하지 못했다.

flag라는 변수를 하나 생성하여 cmd를 shift 시키는게 아니라 flag를 shift 시키고 and 연산하여 출력하려는 숫자 글자수가 5개라하면 첫 번째랑 마지막 번째를 출력하고 싶다면 10001 로 만들어주어 출력하면 될것 같다.

내일아침 다시 구현 계획중.



```
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -a
a option
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -b
b option
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -c
siyun@siyun-CR62-6M:~/my_proj/21$ vi mytest.c
siyun@siyun-CR62-6M:~/my_proj/21$ gcc mytest.c
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -a
a option
siyun@siyun-CR62-6M:~/my_proj/21$ ./a.out -b
b option
siyun@siyun-CR62-6M:~/my_proj/21$ vi mytest.c
```

---파일권한 확인---

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

```
#include <sys/stat.h>
```

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

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

```
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))
```

```
    perm[0] = 'p';
```

```
if(S_ISLNK(buf.st_mode))
```

```
    perm[0] = 'l';
```

```
if(S_ISSOCK(buf.st_mode))
```

```
    perm[0] = 's';
```

```
if(S_ISBLK(buf.st_mode))
```

```
    perm[0] = 'b';
```

```
for(i=0;i<9;i++)
```

//[name . rwx ,rwx ,rwx] 를 채워놓기위해 포문

```
if((buf.st_mode >> (8 -i)) & 1)
```

```
    perm[i+1]=rwx[i%3];
```

```
printf("%s\n",perm);
```

```
return 0;
```

```
}
```

i 가 0일 때

buf를 8번 오른쪽으로 쉬프트 모든칸이 1로 채워져있었다면 8번 쉬프트하면 결과1

1	1	1	1	1	1	1	1	1
8(MSB)	7	6	5	4	3	2	1	0(LSB)
0	0	0	0	0	0	0	0	1
8(MLB)	7	6	5	4	3	2	1	0(LSB)

여기서 1과 앤드하면 결과가 참이다.

따라서 이프문을 통과하고 이프문에서 perm[1]=rwx[0] 이 된다.

다시 1이 들어오면

0	0	0	0	0	0	0	1	1
8(MSB)	7	6	5	4	3	2	1	0(LSB)

3이 되고 3&1은 1이 되어 통과한다.

perm[2] = rwx[1] 이된다.

rwx 들어있는 문자열은 rwx 이며

이걸 다했을 때 perm을 표현하면

perm[i]

x	w	r	x	w	r	x	w	r	state
9MSB)	8	7	6	5	4	3	2	1	0(LSB)

이 되고 buf.mode에서 읽은 state 에 따라 rwx 가 활성화 또는 비활성화 될수 있다.

---setuid---

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

```
#include <sys/stat.h>
```

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

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

```
#include <pwd.h>
```

```
#include <grp.h>
```

```
#include <time.h>
```

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

```
{
```

```
    struct stat buf;
```

```
    struct dirent *p;
```

```
    struct passwd *pw;
```

```
    struct group *gr;
```

```
    struct tm *tm;
```

```
    char ch;
```

```
char perm[11] = "-----";
```

```
char rwx[4] = "rwx";
```

```
char sst[4]= "sst";
```

```
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))
```

```
perm[0]='p';
```

```
if(S_ISSOCK(buf.st_mode))
```

```
perm[0]='s';
```

```
if(S_ISCHR(buf.st_mode))
```

```
perm[0]='c';
```

```
if(S_ISBLK(buf.st_mode))
```

```
perm[0]='b';
```

```
for(i=0;i<9;i++)
```

```
if((buf.st_mode >> (8-i))&1)
```

```
perm[i+1] = rwx[i%3];
```

```
for(i=0;i<3;i++)
```

```
if((buf.st_mode >> (11-i)) &1)
```

```
if(perm[(i+1)*3]!='-')
```

```
perm[(i+1)*3]=sst[i]^0x20;// x(실행) 권한 없으면 대 문자 없으면 s셋팅
```

```
else
```

```
perm[(i+1)*3]=sst[i]; //rwx -> rws
```

```
printf("%s",perm);
```

```
printf("%ld",buf.st_nlink); //파일의 개수
```

```
pw=getpwuid(buf.st_uid);
```

```
printf("%s",pw->pw_name); //사용자 컴퓨터 이름표시
```

```
gr=getgrgid(buf.st_gid); //그룹에 이름을 얻어서 밑줄에서 출력
```

```

printf("%s",gr->gr_name);
printf("%ld",buf.st_size);
tm=localtime(&buf.st_mtime);
printf("%d-%02d-%02d      %02d:%02d:",tm->tm_year      +
1900,tm->tm_mon+1, tm->tm_mday,tm->tm_hour, tm->tm_min);
//날자시간출력 (생성날자 같음)
printf("\n");
return 0;
}

```

용어정리

touch - 여러파일을 한번에 생성할 때 사용한다.

ls -i -

```

siyun@siyun-Z20NH-A551B5U:~/my_proj$ ls -i
12323753 16      12191575 a.out      12323392 class10    12323414 class12    12323754 class16
12201322 1.c      12200573 aseembly  12320920 class11    12320776 class15    12323675 class17
siyun@siyun-Z20NH-A551B5U:~/my_proj$

```

ls -l 파일의 리스트를 자세히 보여준다. 이름 권한 그룹이름 개수 날자 기타등등

```

siyun@siyun-Z20NH-A551B5U:~/my_proj/class21$ ls -l
total 64
-rwxr-xr-t 1 siyun siyun 8992 3월 22 15:29 a.out
-rwSr--r-- 1 siyun siyun 0 3월 22 10:04 a.txt
-rw-rw-r-- 1 siyun siyun 0 3월 22 10:04 b.txt
-rw-rw-r-- 1 siyun siyun 0 3월 22 10:04 c.txt
-rw-rw-r-- 1 siyun siyun 0 3월 22 10:04 d.txt
-rw-rw-r-- 1 siyun siyun 0 3월 22 10:04 e.txt
drwxrwxr-x 2 siyun siyun 4096 3월 22 10:04 fire
drwxrwxr-x 2 siyun siyun 4096 3월 22 10:04 getout
drwxrwxr-x 2 siyun siyun 4096 3월 22 10:04 gogo
-rw-rw-r-- 1 siyun siyun 1339 3월 22 11:39 int.c
-rw-rw-r-- 1 siyun siyun 750 3월 22 11:49 ls.c
-rw-rw-r-- 1 siyun siyun 900 3월 22 14:56 lsg2.c
-rw-rw-r-- 1 siyun siyun 1252 3월 22 15:36 lsg3.c
-rw-rw-r-- 1 siyun siyun 672 3월 22 14:47 lsg.c
-rw-rw-r-- 1 siyun siyun 655 3월 22 12:06 lsint3.c
-rw-rw-r-- 1 siyun siyun 553 3월 22 12:09 lsint.c
-rw-rw-r-- 1 siyun siyun 977 3월 22 13:36 lsl.c
-rw-rw-r-- 1 siyun siyun 454 3월 22 10:35 lsmaingetoption.c
-rw-rw-r-- 1 siyun siyun 566 3월 22 10:23 lsoption.c
siyun@siyun-Z20NH-A551B5U:~/my_proj/class21$

```

chmod - 파일의 모드 접근권한을 바꿔주는거 같다.

ls -R 디렉토리 전체를 보여준다.

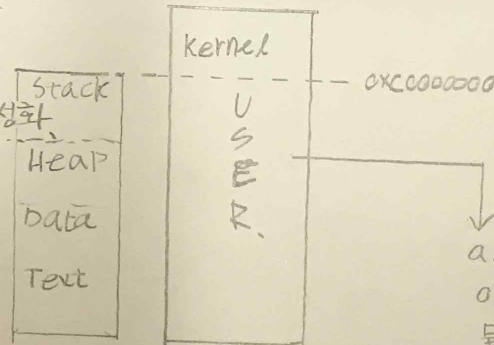
paging and LRU알고리즘

* sticky bit가 붙어있으면 LRU 알고리즘을 한다.

페이징 - 메모리의 진짜 주소를 찾는 방법, 루트

a.out을 실행시키면 프로세스가 된다.

프로세스가 되면 task_struct 가 생성됨
 task_struct
 {
 files_struct
 {
 file
 path
 inode
 super_block
 }
 mm_struct
 Vm_area_struct
 [가상메모리 레이어]
 }



32bit라 가정,

VMI(가상메모리)
 *리눅스에서 메모리 할당
 1:3 = kernel: user

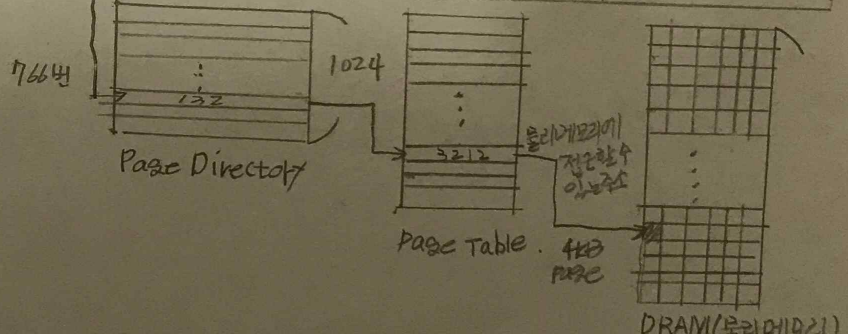
a.out 프로세스의 가상메모리 주소가 0xb0884c8c 라 할 때 물리메모리를 찾기 위해서는 물리메모리의 주소를 갖고있는 가상메모리 주소를 3구간으로 나뉘어야 한다.

우선 2진수 비트단위로 위 주소를 표시하면

1 0 1 1 1 1 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 0

10bit Directory 10bit Table 12bit Offset

766 132 32/2



물리메모리에 저장된 실제 주소를 찾는 방법
 즉 DRAM에 페이지에 들어있는 주소를 찾는
 방법이 페이징.

LRU알고리즘, sticky bit가 붙어있는 프로세스를 처리하는 알고리즘.

* Sticky bit: 용량이 커서 처리하기 위해 붙여줌. (swap해서 포트)

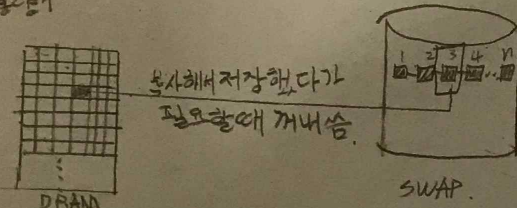
Memory 계층구조

속도↑ Register
 ↓ 용량↓
 Cache
 ↓ 용량↓
 Memory
 ↓ 용량↓
 Disk

메모리는 느려서 Demand on paging을 한다.

필요한 정도만 페이징

하지만 용량이 커서 부족하면 swap의 힘을 빌린다.



SWAP에는 상대적으로 재사용빈도가 적은걸 넣어준다. SWAP에 들어있는 page들은 연결리스트 형태로 있다
 Queue