```
set tags =/root/compiler/gcc-4.5.0/tags
if version >= 500
func! Sts()
	let st = expand("<cword>")
	exe "sts"st

endfunc
nmap ,st : call Sts()<cr>
func! Tj()
	let st = expand("<cword>")
	exe "tj".st
endfunc
nmap,tj :call Tj()<cr>
endif

set csprg=/usr/bin/cscope
set nocsverb
cs add /root/compiler/gcc-4.5.0/cscope.out
set csto = o
set cst
func! Css()
	let css = expand("<cword>")
  new
	exe "cs find s ". css
	if getline(1) ==""
		exe "q"
	endif
		endfunc
nmap, css :call Css()<cr>
func! Csd()
	let csd = expand("<crord>"3
	new
	exe "cs find d".csd
	if getline(1) ==""
		exe "q!"
	endif
	endfunc
	nmap, csd : call Csd() <cr)

func! Csg()
	let csg =expand("<cword>")
	new
	exe "cs find g".csg
	if getline(1)==""
	  exe "q!"
	endif
endfunc
nmap,csg :call Csg()<cr>
```

```
#!/bin/sh
rm -rf cscope.files.files cscope.files
find . \( -name'*.c'-o-name'*.cpp'-o-name'*.cc'-o-name'*.h'-o-name'*.S'\)-print >cscope.files
cscope -i cscope.files
```

chmod 755 ~/mkcscope.sh //초록

sudo ~/mkcscope.sh /usr/local/bin/ //어디서든 mkc 호출가능

리눅스 커널설정 cscope 설정 **mkcsope.sh** 설정

vi -t task_struct

커서위에 가서 컨트롤 대괄호
ctrl +] = 커서위치의 정보찾기

하나 했던거 되돌리는건 **ctrl + T**

앞에 아키가 붙으면 다 지움 볼필요없음

드라이버는 하드를 동작시키는 일종의 펌웨어 장치를 구동시키는 드라이버
드라이버도 다 제낌

fs = file system 의 약자 **(** 파일자체를 보기때문에 제낌

ar , 드라이버 **, fs** 는 안봐도됨
include 를 봐야됨 **(** 구조체 선언 **{**이 거기있음 **)**

include 152 번에 구조체 시작점 큐를 누르고 **152** 번

한페이지 넘기기는 **space** 키

지금하는건 **c tag, cscope**

3 개의 파일을 압축하는건

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

typedef struct
{
	char fname[20];
	int fsize;
}F_info;

int file_size(int fd)
{
	int fsize, old;
	old = lseek(fd, 0, SEEK_CUR);
	fsize = lseek(fd, 0, SEEK_END);		//전체 파일의 사이즈가 결정
	lseek(fd, old, SEEK_SET);		//원상복구
	return fsize;				//파일전체 사이즈가 리턴
}
int main(int argc, char *argv[])
{
	int src, dst, ret;
	char buf[104];
	F_info info;				//파일의 네임
	int i;
	dst = open(argv[argc -1], O_WRONLY|O_CREAT|O_TRUNC<0644);
	for(i =0; i <argc -2; i++)			//맨마지막 꺼 전까지 돌고있음
	{
		src = open(argv[i + 1], O_RDONLY);
		strcpy(info.fname, argv[i + 1]);			//파일의 이름을 info.fname 에 복사
		info.fsize = file_size(src);
		write(dst, &info, sizeof(info));		//info 구조체의 주소를 전달
		while(ret = read(src, buf, sizeof(buf)))			//ret 에 src(a.txt)에서 읽는 buf 에 넣음
			write(dst,buf, ret);
		close(src);
	}
	close(dst);
	return 0;
}
```

a.txt 랑 b.txt 랑 c.txt 만들고

./a.out a.txt b.txt c.txt res.tar 하면 세개가 res.tar 이라는 곳에 파일이 하나로 묶임

ls 로 보면 빨간색으로 res.tar 이 생긴걸 볼수있음

./a.out
./a.out res.tar 이게 압축해제

rm -rf a.txt b.txt c.txr 지우는 거

퀴즈`**1**

임의의 난수를 발생시켜서 이값을 배열에 저장하고 **(**중복은 안됨**)**
배열에 저장된 값을 파일에 기록한다
그리고 이값을 읽어서 **Queue** 를 만든다
이후에 여기 저장된 값중 짝수만 선별하여 더한 후에 더한값을 파일에 저장하고
저장한 파일을 읽어 저장된 값을 출력한다

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

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

int extract_idx;

typedef struct __queue
{
	int data;
	struct __queue *link;
} queue;

bool is_dup(int *arr, int cur_idx)
{
	int i, tmp = arr[cur_idx];

	for(i = 0; i < cur_idx; i++)		// i 가 인자 cur_idx 로 오고  cur_idx 가 커지면서
		if(tmp == arr[i])		// tmp 가 arr[i]랑 같아지면 true 리턴 아니면 false
			return true;

	return false;
}

void init_rand_arr(int *arr, int size)			//난수발생함수
{
	int i;

	for(i = 0; i < size; i++)
	{
redo:
		arr[i] = rand() % 10 + 1;			// 난수를 arr[i]에 넣기

		if(is_dup(arr, i))				// 함수
		{
```

```
                        printf("%d dup! redo rand()\n", arr[i]);  //출력
                        goto redo;                                          // goto 문 redo 로
                }
        }
}
/*


int main(void)
{
        int arr[100] = {0};
        int i,j;


        for(i =0; i<10; i++)
        {
                arr[i] = rand()%100;  // 0~ 99
                if(arr[i] == arr[i+1])
                        {
                        i-=i-1;

                        }

        printf("%d\n", arr[i]);
        }
        return 0;


*/


void print_arr(int *arr, int size)                              //출력함수
{
        int i;

        for(i = 0; i < size; i++)
                printf("arr[%d] = %d\n", i, arr[i]);                    // i 가 size 가 될때까지 출력
}

queue *get_queue_node(void)


{
        queue *tmp;
        tmp = (queue *)malloc(sizeof(queue));
        tmp->link = NULL;
        return tmp;
}

void enqueue(queue **head, int data)
{
        if(*head == NULL)
        {
                *head = get_queue_node();
                (*head)->data = data;
                return;
        }

        enqueue(&(*head)->link, data);
```

```
}

void extract_even(queue *head, int *extract)
{
	queue *tmp = head;

	while(tmp)
	{
		if(!(tmp->data % 2))
			extract[extract_idx++] = tmp->data;
		tmp = tmp->link;
	}
}

int main(void)
{
	int i, fd, len, sum = 0;
	char *convert[10] = {0};
	int arr[11] = {0};
	char tmp[32] = {0};
	int extract[11] = {0};
	int size = sizeof(arr) / sizeof(int) - 1;
	queue *head = NULL;

	srand(time(NULL));

	init_rand_arr(arr, size);
	print_arr(arr, size);

	for(i = 0; i < size; i++)
		enqueue(&head, arr[i]);

	extract_even(head, extract);
	printf("\nExtract:\n");
	print_arr(extract, extract_idx);

	fd = open("log.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644);  //  log.txt 파일을 만듬 쓰기모드로 혹시 안에 내용이있다면 다지움
	 //O_RDONLY- 읽기모드 O_WRONLY- 쓰기모드
	//O_RDWR -읽기/쓰기모드
	// O_CREAT -해당파일이 없는 경우에 생성함 , mode 는 생성할 파일의 사용권한을 나타냄
	//O_TRUNC -파일이 이미 있는 경우 내용을 지움


	for(i = 0; i < extract_idx; i++)
		sum += extract[i];

	sprintf(tmp, "%d", sum);
	write(fd, tmp, strlen(tmp));                    //tmq 부터 strlen(tmp)까지 문자열을 씀
	close(fd);

#if 0
	for(i = 0; i < extract_idx; i++)
	{
		int len;
		char tmp[32] = {0};

		sprintf(tmp, "%d", extract[i]);
```

```
                len = strlen(tmp);
                convert[i] = (char *)malloc(len + 1);
                strcpy(convert[i], tmp);
                printf("tmp = %s\n", tmp);
        }
#endif

        return 0;
}
```

퀴즈 **2**

카페에 있는 **50** 번 문제 **(**성적관리**)**을 개조한다
기존에는 입력 받고 저장한 정보가 프로그램이 종료되면 날아갔다**.**
입력한 정보를 영구히 유지할수 있는 방식으로 만들면 더 좋지 않을까**?**

*****조건
1.파일을 읽어서 이름정보와 성적 정보를 가져온다**.**
2. 초기 구동시 파일이 없을 수 있는데 이런경우엔 읽어서 가져올 정보가 없다**.**
3. 학생 이름과 성적을 입력할 수 있도록 한다**.**
4. 입력된 이름과 성적은 파일에 저장되어야 한다**.**
5. 당연히 통계 관리도 되어야한다**(**평균**,** 표준편차**)**
6.프로그램을 종료하고 다시키면 이제 파일에서 앞서 만든 정보들을 읽어와서 내용을 출력해줘야 한다**.**
7.언제든 우너하면 내용을 출력할수 있는 출력함수를 만든다**.**
[특정버튼을 입력하면 출력이 되게 만듬**]**
(역시 **system Call** 기반으로 구현하도록 함**)**