

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

강사 – Innova Lee( 이상훈 )

[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 – 장성환

[redmk1025@gmail.com](mailto:redmk1025@gmail.com)

vi ~/.vimrc  
오른쪽 파일 작성

vi ~/mkcscope.sh  
오른쪽 파일 작성

해당 커널파일 다운로드 받은곳 들어가서 (cd linux-4.4)  
sude apt-get install ctags cscope

ctags -R 입력

Chmod 755 ~/mkcscope.sh

mkcscope 파일을 초록색으로 변경

linux 4.4 폴더에서  
sudo cp ~/mkcscope.sh /usr/local/bin/ 눌러줌 - bp

vi -t task\_struct 로 검색가능

144  
/files\_struct  
ctrl + ] - 해당 문구가 쓰이는곳 모두 찾기.  
구조체를 보기 위해서 해당 이름 { 을 찾는다.

driver - HW 를 구동 시키는 펌웨어 개념?  
fs - 파일 시스템의 약자  
include - 공통적으로 활용하는 부분  
따라서 include 를 유심히 보면 찾을 수 있다.

tap 누르면 해당 파일을 다 안쓰고도 입력이 된다.

vi ~/.vimrc

set ts=8  
set sw=4  
set sts=4  
set smartindent  
set cindent

"ctags 설정"  
set tags=/root/compiler/gcc-4.5.0/tags  
(bp 이후에 set tags=/home/sunghwan/kernel/linux-4.4/tags 로 바꿔준다.)

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

"cscope 설정"  
set csprg=/usr/bin/cscope  
set nocsverb  
cs add /root/compiler/gcc-4.5.0/cscope.out  
(bp 이후에 cs add /home/sunghwan/kernel/linux-4.4/cscope.out 로 바꿔준다.)  
set cst=0  
set cst  
func! Cst()  
    let cst = expand("<cword>")

:w 저장하기

뒤로나가기 ctrl + t

한 페이지씩 넘기기 space

```
new
exe "cs find s ".css
if getline(1)==""
    exe "q!"
endif
endfunc
nmap ,css :call Css(<cr>

func! Csc()
    let csc = expand("<word>")
    new
    exe "cs find c ".csc
    if getline(1) == ""
        exe "q!"
    endif
endfunc
nmap ,csc :call Csc(<cr>

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

vi ~/mkcscope.sh

#!/bin/sh
rm -rf cscope.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
```

윈도우 ntsp 와 fat32 파일 시스템 2 개  
리눅스는 1000 개 이상의 파일 시스템을 다룰 수 있다.

파일 시스템이 다르다는 것은  
어떤 파일을 저장할 때, 포맷이 다르다는 것이다.  
각각이 다른 함수 및 인자를 넣어서 만들기 보다는  
주소만 전달해 주면 알아서 그것에 맞는 함수 포인터를 이용하여 처리..

\* tar\_imp.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.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);
    lseek(fd, old, SEEK_SET); // return to initial cursor (move cursor)
    return fsize;
}
```

a.txt 는 hello  
b.txt 는 linux system  
c.txt 는 system call 입력 후  
./a.out a.txt b.txt c.txt res.tar 실행 후  
xxd res.tar 입력하면 해당 파일의 바이너리 값이 나온다.

lseek()

파일의 임의의 위치로 읽기/쓰기 포인터 이동 함수

http://tseum.falinux.com/character.php?document\_id=222040 파일 및 디렉터리 다루기 99% 2009.07.05 11:31:56 P. 35, 127, 160

**설명**  
파일의 임의(쓰기) 위치를 파일의 처음 위치로 초기화합니다.  
파일의 위치를, 다음 조건에 따라 앞으로 또는 뒤로 임의(쓰기) 위치로 건너 뛸니다.

whence	설명
SEEK_SET	파일의 시작
SEEK_CUR	현재 임의(쓰기) 포인터 위치
SEEK_END	파일의 끝

건너 뛸 때는 정렬된 변수로 받은 숫자에 위치를 이동한다는 뜻이 아니라 건너 제곱의 count 한다는 뜻입니다. 아래에 그림을 참고 하여 주십시오.

\* 그림은 fseek()에 사용된 그림입니다. 그러나 whence가 의미하는 내용은 끝으로 그대로 사용하였습니다.  
\* fseek()는 FILE "fp"를 사용했고 lseek()는 int fd를 사용한 것에 주의하세요.

첫 번째 fseek()는 앞의 그림 이후 위치를 반환하므로 SEEK\_END를 이용하여 파일 크기를 구할 수 있습니다. 즉,  
sz\_file = fseek( fd, 0, SEEK\_END);  
SEEK\_END에서 0 위치로 해당되는 읽기 쓰기 포인터의 위치가 곧 파일의 끝이므로 파일 크기를 구할 수 있습니다.

**예제**  
unistd.h  
fseek

**형태** off\_t fseek(int fd, off\_t offset, int whence);

**인수** int fd는 파일 디스크립터  
off\_t offset 이동할 바이트의 수  
int whence 시작 위치

```
int main(int argc, char *argv[]){
    int src, dst, ret;
    char buf[1024];
    F_info info;
    int i;
    dst = open(argv[argc - 1], O_WRONLY | O_CREAT | O_TRUNC,
0644);
    //last file
    for(i=0; i<argc-2;i++){
        src = open(argv[i+1], O_RDONLY); // except excute file
        strcpy(info.fname, argv[i + 1]); // fname = files name
        info.fsize = file_size(src);
        write(dst, &info, sizeof(info));
        while(ret = read(src, buf, sizeof(buf)))
            write(dst, buf, ret);
        close(src);
    }
    close(dst);
    return 0;
}
```

\*tar\_free.c

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

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

```
#define min(x,y) (((x)<(y)) ? (x):(y))
```

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

```
    int src, dst, len, ret;
```

```
    F_info info;
```

```
    char buf[1024];
```

```
    src = open(argv[1], O_RDONLY);
```

```
    while(read(src, &info, sizeof(info))){
```

```
        dst = open(info.fname, O_WRONLY | O_TRUNC | O_CREAT,
0644);
```

```
        while(info.fsize > 0){
```

```
            len = min(sizeof(buf), info.fsize);
```

```
            ret = read(src, buf, len);
```

```
            write(dst, buf, ret);
```

```
            info.fsize -= ret;
```

```
        }
```

```
        close(dst);
```

```
    }
```

```
    close(src);
```

```
    return 0;
```

압축된 파일을 해제하는 기법

}	
---	--

<pre> #include &lt;stdio.h&gt; #include &lt;fcntl.h&gt; #include &lt;unistd.h&gt;  int main (int argc, char **argv){      char buff[1024];     int fd[2];      fd[0]= open("tar_free.c",O_RDONLY);     read(fd[0],buff,10);     write(1,buff,10);      fd[1]= open("tar_free.c",O_RDONLY);     read(fd[1],buff,10);     write(1,buff,10);      return 0; } </pre>	<p>*리눅스를 할때 핵심</p> <p>task_struct 이게 실행 시키는 프로그램이라고 생각하면 된다.</p> <pre> task_struct{ files_struct *  files_struct{ files* </pre> <p>여기서 files *에 파일을 받게 되는데 open 을 하면 별도의 파일 포인터가 생성이 되어서 실행할 때 다른 포인터로 받아서 실행해도 같은 값이 나오게 된다.</p>
---	---



\* question 1

의외의 난수를 발생시켜서 이 값을 배열에 저장하고  
배열에 저장된 값을 파일에 기록한다.

(중복은 안됨)

그리고 이 값을 읽어서 queue 를 만든다.

이후에 여기 저장된 값 중 짝수만 선별하여 모두 더한 후에  
더한 값을 파일에 저장하고 저장된 값을 출력하도록 한다.

Man sprintf 누르면 함수의 정보가 나온다.

\* quiz1\_1.c 를 참조

\* question 2

카페에 있는 50 번 문제를 개조한다.

어떻게 개조할 것인가 ?

기존에는 입력 받고 저장한 정보가 프로그램이 종료되면 날아갔다.

입력한 정보를 영구히 유지 할 수 있는 방식으로 만들면 좋지 않을까?

\*조건

1.파일을 읽어서 이름 정보와 성적 정보를 가져온다.

2. 초기 구동시 파일이 없을 수 있는데 이런 경우엔 읽어서 가져올 정보가 없다.

3. 학생 이름과 성적을 입력할 수 있도록 한다.

4. 입력한 이름과 성적은 파일에 저장되어야 한다.

5. 당연히 통계 관리도 되어야 한다(평균, 표준 편차)

6. 프로그램을 종료하고 다시 키면 파일에서 앞서 만든 정보들을 읽어와서 내용을 출력해줘야 한다.

7. 언제든지 원하면 내용을 출력할 수 있는 풀력함수를 만든다.

[특정 버튼을 입력하면 출력이 되게 만듦]

(역시 시스템 콜 기반으로 구현하도록 함)

\*ans\_quiz2.c 참조

내가 만든건

\*quiz2.c 참조