TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

19 일차
리눅스 프로그래밍 2 일차

윈도우가 지원하는 파일시스템은 FAT, NTFS 만 즉, 자신것들만 지원

운영체제 설치할때 리눅스는 파일시스템이 있어서 윈도우가 설치되있는지 알지만
윈도우는 우분투가 설치되있는지 모름
리눅스가 지원하는 파일시스템이 1000 개

함수포인터를 사용하는이유
open 을 할때 그 파일에맞는 주솟값이들어가서 열려짐

/* open() 을 통해서 얻게 되는 File Descriptor 의 번호는
결국 이 배열의 인덱스에 해당한다.
커널은 별도의 정보를 제공하지 않고
이 인덱스 정보만을 제공하므로
시스템 내부에 치명적인 손상을 줄 수 있는
포인터 주소등을 주지 않고도 유저가 파일을 제어할 수 있게 해줌
그래서 read, write, close 등에는 숫자만 전달하게됨
이 요청을 커널이 받으면 숫자값을 보고
어떤 파일을 제어해야 하는지 빠르게 파악할 수 있음*/

```
#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); //old 값을 구한건 어디로 원상복구시킬지를알기위해
        fsize = lseek(fd, 0, SEEK_END);
        lseek(fd,old,SEEK_SET); //fd 를 올드로 설정, 파일을 원상복구시켜라
        return fsize; /파일전체사이즈가 반환
}
```

```
int main(int argc, char *argv[])
{
        int src, dst, ret;
        char buf[1024];
        F_info info;
```

```
	int i;
	dst = open(argv[argc -1//가장마지막에 들어온인자 .배열에시작은 0 이기때문에, 맨마지막에는 모든
파일을 묶을 파일명을 적어주면됨], O_WRONLY | O_CREAT |	O_TRUNC, 0644);
	for(i=0; i<argc–2; i++) //for 문도는데 argc-2 까지 돔 , 맨마지막꺼전까지 돌기위해
	{
		src = open(argv[i+1],O_RDONLY);//+1 인이유는 실행파일을 집어넣으면안되기때문에
실행파일은 binary 파일
		strcpy(info.fname,argv[i+1]); //cp 는 info.name 에 argv 파일의 이름을 복사
		F_info
		info.fsize =file_size(src);
		write(dst,&info,sizeof(info)); //info 구조체 주소를 전달하고있음 . Res.tar 의 내용을
							info 에 쓰겠다
		while(ret=read(src,	buf,sizeof(buf))) //ret 는 리드하고있음 src 에서 내용을 읽고있음
			a.txt 에 있는 내용을 읽어서 버퍼에 넣음 버퍼에 놓고 읽은 바이트의수가 ret 에 감
			write(dst,buf,ret);
		close(src);
	}
	close(dst);
	return 0;
}
```

실습
a.txt 에는 hello 를 넣고
b.txt 에는 Linux System 을 넣고
c.txt 에는 System Call 을 적고

/mytar a.txt
b.txt	c.txt	res.tar
명령어넣으면 동작

→ xxd res.tar
결과로 16 진수들보임

묶어놓은걸 해제하는 tar_free
tar_free lmplemantation

```
#include<fcntl.h>

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

#define min(x,y)	(((x) < (y)) ? (x) : (y)) //x 는참 y 는거짓
```
x 에 복잡한수식이 들어올걸 대비해서 괄호가 많이배치됨

```
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))) //info 구조체를 읽어들이겠다는거
	{
		dst = open(info.fname, O_WRONLY | O_TRUNC | O_CREAT, 0644);
		while(info.fsize > 0)
		{
			len = min(sizeof(buf), info.fsize); //fsize 가 1024 를 넘어갈수도있음
넘지않는다면 넘지않는값만큼만 사용하기위해 min 이 있음
			ret = read(src, buf, len); //src 는 res.tar
			write(dst, buf, ret);
			info.fsize -= ret;
		}
		close(dst);
	}
	close(src);
	return 0;
}
```

rm -rf a.txt b.txt c.txt
res.tar 만 남기기
./

tar2.c

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

int main(void)
{
	char buff[1024];
	int fd[2];

	fd[0] = open("mytal.c",O_RDONLY);
	read(fd[0], buff, 10);
	write(1, buff, 10);

	fd[1] = open("mytar.c",O_RDONLY);
	read(fd[1],buff,10);
	write(1,buff,10);
	return 0;
}
```

리눅스를 하게될때 핵심
프로그램
task_struct

files_struct * → files_struct
ㄴ file * 오른쪽화살표 file [4]
→file [3] //오픈을하면 별도의 파일디스크랩터가 뚫린다는것

퀴즈
Quiz. 1
임의의 난수를 10 개 발생시켜서 이 값을 배열에 저장하고
배열에 저장된 값을 파일에 기록한다. (중복은 안됨)
그리고 이 값을 읽어서 Queue 를 만든다.
이후에 여기 저장된 값중 짝수만 선별하여 모두 더한 후에
더한 값을 파일에 저장하고 저장된 값을 출력하도록 한다.
(반드시 System Call 기반으로 구현하도록 함 - 성능이 압도적임)

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

int main(void)
{
	srand(time(NULL));

	int random = rand() %10;

	printf("%d\n",random);


	return 0;
}
```

어떤함수를 쓰고싶은데 사용법이 기억이 안난다면
man sprintf 들어가서 디스크립션을 확인

sum += extract[i] // 숫자로 계산
sprintf(tmp, “%d”, sum); //숫자를 문자값으로 받는다
write(fd, tmp) //문자로 받은걸 write 로 쓴다

quiz1_1.c

```
#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 //typedef 선언
{
	int data; //int 형 변수 data
	struct __queue *link; //링크
} queue;

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

	for(i = 0; i < cur_idx; i++)
		if(tmp == arr[i])
			return true;

	return false;
}

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

	for(i = 0; i < size; i++) //i 가 size 보다 작다면
	{
redo:
		arr[i] = rand() % 10 + 1;

		if(is_dup(arr, i))
		{
			printf("%d dup! redo rand()\n", arr[i]);
			goto redo;
		}
	}
}

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

	for(i = 0; i < size; i++) //i 가 size 보다 작다면
		printf("arr[%d] = %d\n", i, arr[i]);
```

```
}

queue *get_queue_node(void)
{
	queue *tmp;
	tmp = (queue *)malloc(sizeof(queue)); //멀록으로 큐에대한 메모리(힙)할당
	tmp->link = NULL;
	return tmp; //tmp 값 반환
}

void enqueue(queue **head, int data)
{
	if(*head == NULL) //*head 값이 널이라면
	{
		*head = get_queue_node(); //겟노드로 이동
		(*head)->data = data; //head 가 가리키는 데이타의 값을 데이타에 넣는다
		return; //끝
	}

	enqueue(&(*head)->link, data); //*헤드가 가리키는 링크, 데이타값의 주솟값을
}

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)); //랜덤숫자

	init_rand_arr(arr, size);
	print_arr(arr, size); //arr 의 사이즈를 프린트

	for(i = 0; i < size; i++) //i 의 값이 size 보다 작다면
		enqueue(&head, arr[i]);
```

```
	extract_even(head, extract);
	printf("\nExtract:\n"); // \nExtract:\n 출력
	print_arr(extract, extract_idx); //배열 출력

	fd = open("log.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644); //쓰기전용으로 열기

	for(i = 0; i < extract_idx; i++) //i 의 값이 extract_idx 보다 작다면
		sum += extract[i]; //숫자로 계산

	sprintf(tmp, "%d", sum); sum 만큼의 숫자를 문자값으로 받는다
	write(fd, tmp, strlen(tmp)); // 문자로 받은걸 write 로 쓴
	close(fd);

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

		sprintf(tmp, "%d", extract[i]); // extract[i]만큼의 숫자를 문자로 받는다
		len = strlen(tmp); //받은 문자를 len 에 저장한다
		convert[i] = (char *)malloc(len + 1); //convert[i]에 대한 힙값을 +1 만큼 추가한다
		strcpy(convert[i], tmp); //convert[i]에 받은 문자를 복사
		printf("tmp = %s\n", tmp); //받은 문자를 출력
	}
#endif

	return 0;
}


sum += extract[i] // 숫자로 계산
sprintf(tmp, “%d”, sum); //숫자를 문자값으로 받는다
write(fd, tmp) //문자로 받은걸 write 다로 쓴다
```