TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

Man open
man 2 open // 2 는 system call 에 대한 섹션번호

//es 1 은 쉘명령어 2 는 시스템콜 라이브러리 3 번은 c 라이브러리

man sprintf

```
 sprintf(filepath,"abc:%s",fname); // 문자열을  조립해서 filepath 에  저장
```

파일을 특정한 file descripotion

dup

리 다이렉션
cat mytar.c > ccc

cat < ccc
cat ccc

cat < ccc > ddd

pipe Communication

ps 명령어는 process 의 state 를 보는 명령어에 해당한다

ps -ef | grep bash | grep -v grep | awk ‘{print $2}’

-v grep = bash 를 찾는 프로세스는 제외하라는 뜻

$2 는 PID 보는거 = 인간으로 치면 주민번호
 프로세스 id 값

tail command

tail 명령어는 파일 내용의 마지막부터 읽을때 주로 사용

tail -c 20 mytar.c
tail -n 1/var/log/messages
tail -n 10 1.c >>> 요로케 써도됨
-c 은 문자갯수 ~
-n 은 라인숫자~

CPU

프로세스는 CPU 의 추상화

프로세스가 cpu 의 작업을 얻기위해
context switching 적용이됫고
중요도에 따라 우선순위 생김

CPU 가 여러개있는데 어떻게 전부 동시에 실행되는거지~?

정말 동시야 ? ㅡ 싱글코어
병렬처리는 뭐지 ? ㅡ 멀티코어

컴퓨터구조론
CPU 는 오로지 한 순간에 한가지의 연산만 수행한다
커넥트 스위치

아주 빠른속도로 여러 프로서세들 제어어
cxt switching - > multi-tasking 가능

//////////////서로 통신할수있음

블로킹 vs non 블로킹

~~~~~~~~~~~~~~~~~블로킹

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

int main(void)                    //수행하고 다른 terminal 에서 cat > myfifo 를 수행함
{
	int fd, ret;
	char buf[1024];
	mkfifo("myfifo");
	fd =open("myfifo",O_RDWR);          //서로 연결되서 통신할수있음 myfifo 를
열음
	for(;;)
	{
		ret = read(0, buf, sizeof(buf));              //키보드 입력을 받겠다
		buf[ret - 1] = 0;                 //블로킹
		printf("Keyboard input : [%s}\n", buf);          //버프를 출력
		read(fd, buf, sizeof(buf));
		buf[ret -1] = 0;                  //블로킹
		printf("Pipe input : [%s]\n", buf);          //파이프 출력
	}
	return 0;
}
```
~~~~~~~~~~~~~~~~~

그냥 파일은 -rw-rw 아무것도없음
prw = pipe

crw = 캐릭터 디바이스 // c 순서가있음 . 반드시 순서가 지켜져야하는것들은

캐릭터 디바이스 ex)키보드, 모니터,비디오

brw = 블럭 디바이스   //b 특정단위를 가지고 움직임 2^12 승 4096 바이트

4kb 가 물리메모리의 최소단위 하드디스크도 4k 단위

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~논블로킹

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

int main(void)
{

	int fd, ret;
	char buf[1024];
	fd = open("myfifo", O_RDWR);
	fcntl(0, F_SETFL, O_NONBLOCK);		//F_SETF 권한을 셋팅
				O_NONBLOCK  0 번을 논블럭으로 셋팅

	fcntl(fd, F_SETFL, O_NONBLOCK); // fd 의 읽기와 쓰기를 논블럭으로 설정
	for(;;)
	{
		if((ret = read(0, buf, sizeof(buf))) >0) //읽을게 없으면 그냥 넘어감
		{
			buf[ret -1 ] = 0;
			printf("keyboard input : [%s]\n", buf);
		}
		if((ret = read(fd, buf, sizeof(buf))) > 0) //읽을게 없으면 그냥 넘어감
		{
			buf[ret -1] = 0;
			printf("Pipe input :[%s]\n", buf);
		}
	}
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
	close(fd);
	return 0;
}

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

int main(void)
{
	int fd;
	fd = open("a.txt",O_WRONLY| O_CREAT|O_TRUNC,0644);		//
	close(1);		//1 이 닫히면 출력안됨
	dup(fd);			//종료된것을 복사 ,1 번의 역할일 함
					//3 번인 a.txt 에 기록됨
	printf("출력될까 ? \n");

	return 0;
}




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

int main(void)
{

	int fd;
	char buff[1024];
	fd = open("a.txt", O_RDONLY);
	close(0);
	dup(fd);			//dup 가 하는일이 크로즈한것을 dup 에 들어간
fd 가 대체함
					//파일자체가 0 번을 대체함
	gets(buff);			//a.txt 자체 내용을 입력으로 받아서 버프로
//	printf("출력될까 ?\n");
	printf(buff);			// a.txt 를 글자를 바꿔도 다시 바뀜
```

```
	return 0;				//그래서 버프를 출력을 함
}
```

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

int main(int argc, char *argv[])
{
	int i;
	char ch = 'a';
	int fd = open(argv[1], O_WRONLY|O_CREAT|O_TRUNC,0644);
	lseek(fd, 512 -1, SEEK_SET);			//강제로 512 번째 가져다 놓음
	write(fd, &ch, 1);		//a 가 나뜸
	close(fd);

	//마스터 부트 레코드 라는 곳에서 사용
	// mbr :

	return 0;
}
```