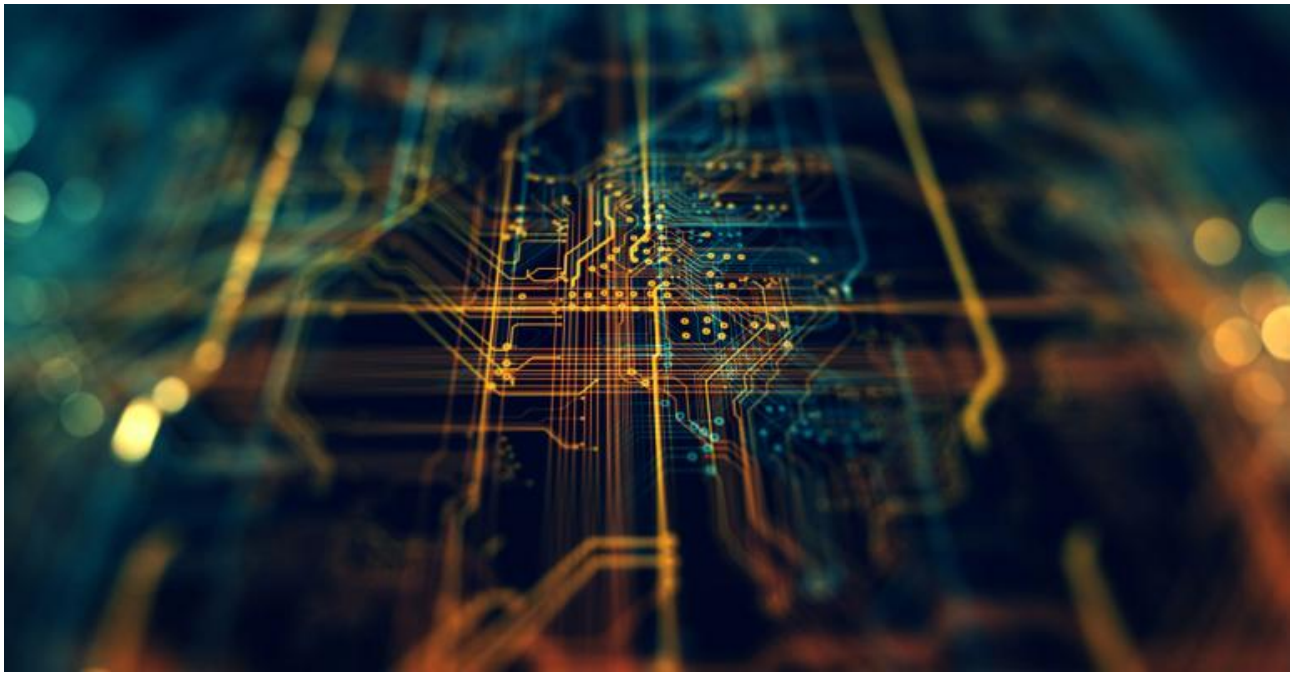
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정



2018.03.26
23 일차
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 신민철
akrn33@naver.com

execve2.c

```
#include<unistd.h>

int main(void)
{
	execlp("ps","ps","-e","-f",0);//메모리레이아웃이 ps 로 바뀌기 때문에 밑에있는 printf 는 안나오게 된다.첫번째 빼고 뒤에것들은 셸옵션
	printf("after\n");//execlp

	return 0;
}
```

execve3.c

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

int main(void)
{
	int status;
	pid_t pid;
	if((pid = fork()) >0 )
	{
		wait(&status);
		printf("prompt > ");
	}
	else if(pid == 0)
		execlp("ps","ps","-e","-f",0);
	return 0;
}
```

//execve2 와 다르게 자식프로세스를 생성해서 하므로 프린트도 출력되고 ps 도 출력된다.

Execve4.c

```
#include<stdio.h>
#include<unistd.h>
int main(void)

{
	int status;
	pid_t pid;
	if((pid = fork()) > 0 )
	{
		wait(&status);
		printf("prompt >");
	}
	else if(pid == 0)
		execl("./newpgm","newpgm","one","two",(char*)0);
		//현재디렉토리(./newpgm 의 newpgm 으로 가서 하나씩 차례로 argv 로들어가라
	return 0;
}
```

execve5.c

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

int main(void)
{
	int status;
	pid_t pid;
```

```
        char* argv[] = {"./newpgm","newpgm","one","two",0};
        char* env[] = {"name = OS_Hacker","age = 20",0};
        if((pid = fork()) > 0)
        {
                wait(&status);
                printf("prompt > \n");
        }
        else if(pid == 0)
                execve("./newpgm", argv, env);
                ./newpgm 으로 가서 캐릭터배열 argv 와 env 를 인자값으로 들
                어가라
        return 0;
}
```

newpgm.c//인자값으로 들어온 것들을 출력하는 code
//execve4.c 와 execve5.c 같이쓰임

```
#include<stdio.h>

int main(int argc, char* argv[],char* env[]){
        int i;
        for( i = 0 ; argv[i]; i++)
        {
                printf("argv[%d] = [%s]\n",i,argv[i]);
                printf("env[%d] = [%s]\n",i,env[i]);
        }
        for(i = 0 ; env[i]; i++)
        return 0;
}
```

execve6.c

```
#include<stdio.h>

int main(void)
```

```
{
	system("date");//이안에서 자식을 만들고 그안에서 이그잭하기때문
	에 date 가 잘 출력됨. date 는 셸.
	printf("after\n");
	return 0;
}
```

execve7.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>

int my_system(char* cmd)
{
	pid_t pid;
	int status;
	char* argv[] = {"sh","-c",cmd,0};//sh 라는 셸에서 -c 는 명령을
실행해라 cmd 라는 명령어를
	char* envp[] = {0};
	if((pid = fork()) > 0)
		wait(&status);
	else if(pid ==0)

		execve("/bin/sh", argv, envp);

}

int main(void)
{
	for(;;)
	my_system("date");
	printf("after\n");
	return 0;
```

```
}
```

execve9.c

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

int daemon_init(void)
{
	int i;
	if(fork() > 0)
		exit(0);
	setsid();
	chdir("/");
	umask(0);
	for(i = 0; i < 64; i++)
		close(i);
	signal(SIGCHLD, SIG_IGN);
	return 0;
}
```

/*sessionid 프로세스가 터미널과 생명을 함께한다는거 패륜을 저지르고 셋에스아이디를 하게되면 chdir 체인지디렉토리 데몬이 구동하다보면 어떤 파일이든지 접근할 필요가 있음 그래서 위치를 루트로 바꿔 주는거임 루트에 있는 모든걸 사용하게 해주겠다가 umask(0)이다. 포문돌면서 모든걸 닫아준다 리눅스에는 64 개의 표준이있어서 그걸 다 닫아주는거다 시그널함수가 나오는건 처리를 하겠단건데 시그차일드가 나왔을때 시그이그노얼은 뭐가 나오든무시하겠다는거*/

```
int main(void)
{
	daemon_init();
```

```
        while(1){
        printf("ns\n");
        }
        return 0;
}
```

exit.c

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<errno.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<fcntl.h>

void term_status(int status)
{
        if(WIFEXITED(status))
                printf("(exit)status : 0x%x\n",WEXITSTATUS(status));
        else if(WTERMSIG(status))
                printf("(signal)status:0x%x,%s\n",
                        status & 0x7f, WCOREDUMP(status)? "core
dumped" : "");
}

int main(void){
        pid_t pid;
        int status;
        if((pid = fork()) > 0)
        {
                wait(&status);
                term_status(status);
        }
        else if(pid == 0)
                abort();
```

```
	else
	{
		perror("fork()");
		exit(-1);
	}
	return 0;
}
```
//이프로그램이시그널마다 죽게되면어느메모리상태에서 죽었는지알아야하는데 그것을 지정해 줄지 말지 정해주는게 코어덤프}

sigit.c

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

int main(void)
{
	signal(SIGINT, SIG_IGN);//컨트롤 c 를 무시하겠다.
	signal(SIGQUIT, SIG_IGN);
	signal(SIGKILL, SIG_IGN);//이건 못막음. Bin/init 을 막을순없음
	pause();
	return 0;
}
```

wait10.c

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

```
#include<sys/wait.h> ㅏ
#include<signal.h>
#include<stdlib.h>

void term_status(int status)
{
	if(WIFEXITED(status))
		printf("(exit)status : 0x%x\n",WEXITSTATUS(status));
	else if(WTERMSIG(status))
		printf("(signal)status : 0x%x, %s\n", status & 0x7f,
WCOREDUMP(status) ? "core dumped" : "");
}

void my_sig(int signo)
{
	int status;
	wait(&status);//시그차일드가 스테이터스로 넘어감
	term_status(status);//정상종료됐는지 확인
}

int main(void)
{
	pid_t pid;
	int i;
	signal(SIGCHLD, my_sig);// 어떤상황에서 어떤 대처를 할 매뉴얼
	if((pid = fork()) > 0 )
	for( i = 0 ; i < 10000; i++)
	{
		usleep(50000);
		printf("%d\n", i + 1);
	}
	else if(pid == 0)
		sleep(5);
	else
```

```
        {
                perror("fork()");
                exit(-1);
        }
        return 0;
}
```