Xilinx Zynq FPGA, TI DSP,MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)

gcccompil3r@gmail.com

학생 - 이유성

dbtjd1102@naver.com

1.비정상 종료.

```c
#include<unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/wait.h>
void term_status(int status)
{
	if(WIFEXITED(status)) //정상 1 비정상 0
		printf("(exit)status : 0x%x\n",WEXITSTATUS(status));
	else if(WTERMSIG(status))//비정상 1 정상 0
		printf("(signal)status : 0x%x, %s\n",status & 0x7f,WCOREDUMP(status) ? "core dumped" :  "");
}   //core dump 비정상 종료 되었을 때, 기록하거나 저장 (1 , 0) 나머지 비트 값은 자식이 죽은 상태 값
	// core dump가 1이면 core dumped 출력
int main(void)
{
	pid_t pid;
	int status;
	if((pid = fork()) >0)
	{
		wait(&status);//자식이 죽을때까지 대기 ..abort 리턴값,
		term_status(status);
	}
	else if(pid ==0)
		abort();//강제종료  자식이 죽으면 부모가 받아,,
	else
	{
		perror("fork() ");
		exit(-1);
	}
	return 0;
}
```

결과값

```
(signal)status : 0x6, core dumped
```

2.signal

```
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <signal.h>
void term_status(int status)
{
	if(WIFEXITED(status)) //정상 1 비정상 0
		printf("(exit)status : 0x%x\n",WEXITSTATUS(status));//정상처리
	else if(WTERMSIG(status))//정상 0 비정상 1
		printf("(signal)status : 0x%x ,%s\n", status & 0x7f, WCOREDUMP(status) ? "core
dumped" : "");
//코어 덤프는 0,1 --코어덤프 말고 나머지 비트는 비정상종료 상태값을 의미  core dumpe가 1이면 core
dumped 출력 0이면  "".
}
void my_sig(int signo){
	int status;
	wait(&status);//sigchld전달  자식의 상태값 기다리는중..
	term_status(status);//비정상으로 죽었는지 정상적으로 죽었는지 구분 하겠다.  자식의 상태에 대한
설명
				//exit에 걸려서 리턴 0 반환, (정상종료)
}
int main(void)
{
	pid_t pid;
	int i;
	signal(SIGCHLD,my_sig); //시그널이라는 함수는 sigchld값이 들어왔을때 my_sig실행 ..  자식이
신호를 보낸다(시그널을 때린다)
기)
				//인터럽트 ,, 시그널은 행동지침을 등록,, 시그널은 매뉴얼 어떤 상황에서 어떤
매뉴얼을 동작 시킬 것인가를 정한다.그래야 대처
// 바로 동작시키지 않는 이유 불이 났을때 처리,상황에 대처..signal..
	if((pid = fork()) >0)
		for( i = 0; i <10000; i++)
		{
			usleep(50000);//us마이크로세컨드 단위 10^-6 , ms10^-3 밀리 세컨드,0.05초
			printf("%d\n",i+1);
		}
	else if(pid ==0)
		sleep(5);
//자식이 죽으면 부모한테 가는데 부모 동작 중..그럼? signal -> my_sig호출.자식이 죽으면
SIGCHLD라는게 생김
	else
	{
		perror("fork() ");
		exit(-1);
	}
	return 0; //정상종료
}
```

결과

카운트 100쯤 될 때 주소값,

//my_sig 날아오니까 동작 (비동기처리)

// 자식이 시그널 맞아 죽은게 아니니까 (정상종료)

3

//wait() = blocking 프로세스 생길떄마다 tast_struct가 많이 쌓임 (좀비 프로세스) ->메모리가 많이 싸임 컴퓨터가 꺼짐,,
//waitpid() wait의 논블로킹,, 프로세스처리 여러개 한번에 들어오면 순번을 정해서 예약..(다 처리하겠단 뜻..)

```
#include<unistd.h>
int main(void)
{
	execlp("ps","ps","-e","-f",0);	//프로그램이름,인자(argument),인자,인자,0(여기가끝)
	return 0;

 //메모리 레이아웃해서 execlp만 빼내와서 이것만 실행 (부모 자식의 개념 x 프로세스 생성 o)
}
```

ps -ef 실행

4

```c
#include<unistd.h>
#include<stdio.h>
int main(void)
{
	execlp("ps","ps","-e","-f",0);//  메모리 레이아웃이 바꼈다. execlp함수만 실행하고 끝낸다.
	printf("after\n"); //레이아웃 : 껍데기 가져가고(stack heap data text)stack에 있는 execlp함수하나 가져간다..
	return 0;
}


//after 출력이 왜 안될까 ?
//프로세스를 새로 갈아타
//fork 자식을만들면서  메모리 복사
```

```
yoosung+  3800  2372  0 20:17 pts/7    00:00:00 ps -ef
```

5

```c
#include<stdio.h>
#include<unistd.h>
int main(void)
{
	int status;
	pid_t pid;
	if((pid =fork()) > 0)
	{
		wait(&status);
		printf("prompt>\n");
	}
	else if(pid == 0) //ps실행하자마자 죽음
		execlp("ps","ps","-e","-f",0);//메모리처리 (이것만 실행하면 처리하고 꺼짐)
	return 0;
}
//2개를 한번에 처리할 수 있따.프로세스 2개 한개는 부모 하나는 자식 자식은 execlp만 수행하고 끝.
```

```
ps -ef 다음에
prompt >
yoosung+  3837  2372  0 20:20 pts/7    00:00:00 ./a.out
yoosung+  3838  3837  0 20:20 pts/7    00:00:00 ps -e -f
prompt>
```

6

```c
#include<unistd.h>
#include<stdio.h>
int main(void)
{
	int status;
	pid_t pid;
	if((pid = fork()) >0)
	{
		wait(&status);
		printf("prompt>\n");
	}
	else if(pid ==0)
		execl("./newpgm","newpgm","one","two",(char *)0);
// 만든newpgm 우리가 만든 프로그램 동시다발적으로 돌리겠다.
                                        //자식은 newpgm을 실행시키겠다(인자 newpgm,one,two)
                                       //char 포인터라는건 NULL을 의미 ;; 대신 NULL을 넣어도 됨.
	return 0;
}
```

newpgm.c

```c
#include<stdio.h>
int main(int argc, char **argv)
{
	int i;
	for(i = 0; argv[i];i++)
		printf("argv[%d] = [%s]\n",i,argv[i]);
	return 0;
}
```

```
argv[0] = [newpgm]
argv[1] = [one]
argv[2] = [two]
prompt>
```

newpgm.c파일을
gcc -g -o newpgm newpgm.c 이후 위의 파일을 컴파일 후 실행.

7

```c
#include<unistd.h>
#include<stdio.h>
int main(void)
{
	int status;
	pid_t pid;
	char *argv[] = {"./newpgm.c","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);
	return 0;
}
```

```
argv[0] = [./newpgm1.c]
argv[1] = [newpgm]
argv[2] = [one]
argv[3] = [two]
envp[0] = [name = OS_Hacker]
envp[1] = [age = 20]
prompt>
```

```c
#include<stdio.h>
int main(int argc, char **argv)
{
    int i;
    for(i = 0; argv[i];i++)
        printf("argv[%d] = [%s]\n",i,argv[i]);
    return 0;
}
```

//에서 envp 인자와 for문을 이용한 프린트만 해주면 된다.

8

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(void)
{
	for(;;)
	{
	sleep(1);
	system("date");//메인프로세스 실행
	}
	printf("after\n");
	return 0;
}
// 이걸 실행시키고 새로운 터미널 창 열고 ps -ef |grep a.out 실행
// 실행한 터미널 지우고 다시 다른 터미널 창에 ps -ef |grep a.out 실행
// ->pts 하나 사라짐을 볼 수 있다.(?나오거나 pts안나오면 재부팅 ㄱㄱ)
//서버는 터미널 끄던 안끄던 살아 있어야한다. ->프로세스는 툭하면 죽어버린다.
//서버를 이렇게 만들면 사고치는거다 . ->대몬프로세스로
```

9

```c
//system과 같은 과정을 만들어본다.
#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};  //문자열을 포인터로 받았다. bash shell 커널 -c 해당 cmd실행 cmd
	char *envp[] = {0};
	if((pid = fork()) >0)
		wait(&status);
	else if(pid ==0)
		execve("/bin/sh",argv,envp);//"경로(시스템 내부) "  배열(v)인자를  보낸다 ,e 환경 변수 설정
}
int main(void)
{
	my_system("date");
	printf("after\n");
	return 0;
}
//결국 시스템에서 포크를 하고 exe을 한다.  포크를해서 프로세스를 만들고 만들어진 프로세스를 exe하면 둔갑을 한다. 그리고 원래 있던 프로세스를 쓸 수 있다.
```

결과

```
./a.out
Mon Mar 26 21:47:08 KST 2018
after
/bin/sh -c date
2018. 03. 26. (월) 21:48:27 KST
```

10

```
//daemon process ==패륜(악마)
// 반드시 만드는 법을 알아야 한다

#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>
int daemon_init(void) //데몬 프로세스 만든다     터미널(콘솔창)
{
	int i;
	if(fork() >0)
		exit(0);  //부모 죽였어.
	setsid(); //tty pts 가상터미널(세션아이디)우리의 프로세스가 터미널과 생명을 같이함
//루트라는 디렉토리로 감 //새로운 세션을 생성한다.
		//종이 악마로 변하면서 tty(세션)에 ?가 생김
	chdir("/"); // 하는이유 ? 대몬을 구동하다보면 어떤 파일에 접근할 필요가 생김
//그래서 위치를 루트로 바꿈
	umask(0); //루트라는 디렉토리로 갔으니까 권한 다 주겠다(안중요함)
	for(i = 0; i <64; i++)// (0 입력 ,1출력 ..파일 64 개 리눅스는 기본적으로 열어둠)
		close(i);//부모와 연을 끊음..
	signal(SIGCHLD,SIG_IGN);//시그널 어떠한 동작을 지침 , sig_ing자식이 죽던말던 신경안쓰고 실행
	return 0;  //안끝남
}
int main(void)
{
	daemon_init();
	for(;;);  //세미콜론..
	return 0;
}
// 데몬이라는 녀석의 생명력이 중요. 쉽게 안죽음 <=> 일반 프로세스는 잘 죽음
// 구글 포털 게임서버 ..이런것들 다 데몬
// 데몬 끊을 수 있는 방법 신?(kill)
```

11

```c
#include<signal.h>
#include<stdio.h>
int main(void)
{
	signal(SIGINT,SIG_IGN);
	signal(SIGQUIT,SIG_IGN);
	signal(SIGKILL,SIG_IGN);
	pause();
	return 0;
}
//signal SIGINT = ctrl+C씹는다
//다막아 놓아도 SIGKILL은 못막음.(신의 철퇴 못막음)
//파일은( 하드디스크의 추상화) <==> 프로세스(cpu의 추상화)와는 다르다
//kill -9맞으면 죽음 (다른터미널)
```