TI DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)
Gccompil3r@gmail.com
학생-윤지완
Yoonjw7894@naver.com

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<errno.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/wait.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();//signal abort
	else
	{
		perror("fork()");
		exit(-1);
	}
	return 0;
}
```

core dump:이 프로그램이 비정상을 끝났을때 어느 메모리에서 끝났는지 그것을 dump 에서 알려줄지 안알려줄지 1:core dump 0:no core dump

<비동기 처리>

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<errno.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/wait.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);
trem_status(status);
}



int main(void)
{
      pid_t pid;
	int i;
	signal(SIGCHLD,my_sig);//행동지침을 점목시킴,이게 없으면 my_sig 가 동작하지 않음
signal 은 언제 어디서든 갑작스런 상황을 대비하는 메뉴얼이다.
	if((pid = fork())>0)

	   for(i=0;i<10000;i++)
{
usleep(50000);
printf("%d\n",i+1);

	}
```

```
    else if(pid == 0)//자식은 만들어지자마자 5 초동안 잔다.
        Sleep(5);//5 초후에 자식이 죽으면 부모한테 가는데 이때 부모도 일을 하고 있고 코드에서 자식이 죽
으면 my_sig 가 호출이 되고 status 값을 wait 에게 넘겨주고 term_status 에게 값이 전달되고 자식은 signal 에
게 죽은 것이 아니라.시간이 다되서 죽은거라 exit 가 동작을 하게 되고 0 이 출력이 된다.이  0 의 값은 return 0 에
값이며 이것은 정상종료가 되었다고 알려주는 것이다.
    else
    {
        perror("fork()");
        exit(-1);
    }
    return 0;
}
```

while(wait(&status)>0);//wait=blocking->이런식으로 자식 노드가 죽지않고 좀비 보드가 되면 동작은 하지않지만 메모리를 차지하게 되고 이것은 cpu 를 뻗게 한다.

wait 의 blocking 동작 코드
while(waitpid(-1,&status,WNOCHANG)>0)
:wait 의 nonblocking 방식 코드
waitpid 가 되고 죽은 자식이 없으면 그냥 빠져나오지만 죽은 자식이 있으면 자식 노드들을 예약시키고 동작이 없을때 그것을 처리한다.

```
int main(void)
{
execlp("ps","ps","-e","-f",0);
//처음 ps = 명령어 나머지는 인자, 0 은 return 0
return 0;
}
```

```
int main(void)
{
execlp("ps","ps","-e","-f",0);//메모리 laout 을 ps 로 바꾸면 ps 안에는 after 가 없기
때문에 printf("after)가 씹혀버린다.즉 메모리가 ps 로 둔갑해 버린다.
printf("after\n");
return 0;
}
```

```
int main(int argc,char **argv, char **envp)
{
	int i;
	for(i=0;argv[i];i++)
	for(i=0;envp[i];i++)
printf(“envp[%d] = %s”, I,envp[i]

}
```

gcc -o newpgm test.c(파일 생성)

```
#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);
	}
	return 0;

}
```

```
#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};
char *envp[] = {0};
if((pid = fork())>0)
wait(&status);
else if(pid ==0)
execve("/bin/sh",argv,envp);
}
```

```
int main(void)
{
my_systrm(“data”);
printf("after\n");
return 0;
}

#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);
        }
        return 0;

}
```

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

```
}


int main(void)
{
daemon_init();
sleep(20);
return 0;
}
```

<데몬을 죽이지 못하는 코드>

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

int main(void)
{
siganl(SIGINT, SIG_IGN);//SIGINT = CTRL+C 를 무시한다.
siganl(SIGQUIT, SIG_IGN);
siganl(SIGKILL, SIG_IGN);

pause();
return 0;
}
```