Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

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

/*
void term_status(int status)
{
    if(WIFEXITED(status)) // 정상 종료
        printf("(exit)status : 0x%x\n", WEXITSTATUS(status));
    else if(WTERMSIG(status)) // 비정상 종료
        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);//자식 프로세스 작업이 끝날 때 까지 대기, 자식 프로세스가 종료한 상태를 구함
        term_status(status);
    }
    else if(pid == 0)
        abort();//abort 는 signal 을 보낸것이다. 6번
    else
    {
        perror("fork() ");
        exit(-1);
    }
    return 0;
}
*/
```

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

}

void my_sig(int signo) // sigchild 전달
{
    int status;
    wait(&status);
    term_status(status);
}

int main()
{
    pid_t pid;
    int i;
    signal(SIGCHLD, my_sig); // sigchld 가 나올 때 my_sig 를 실행하라.자식 프로세스가 죽으면 무조건 실행 ...
    if((pid = fork()) > 0)
        for(i = 0; i < 1000; i++)
        {
            usleep(50000); // 0.05초 간격으로 조건문 실행
            printf("%d\n", i + 1);
        }
    else if(pid == 0)
        sleep(5); // 5초후에 my_sig 함수 실행
    else
    {
        perror("fork() ");
        exit(-1);
    }
    return 0;
}
```

```c
int main()
{
    execlp("ps","ps","-e","-f",0);//첫 ps는 명령어, 그 이후엔 인자(인덱스) 0은 널 값
    return 0;
}
*/

/*
int main()
{
    execlp("ps","ps","-e","-f",0);//둔갑술 ? 형태가 바뀐다. 메모리에 있는 형태가 바뀐다. 메모리를 ps 로 바꾼다. 메모리 레이아웃이 ps로 변경
    printf("after\n");//실행안됨.
    return 0;
}
*/
```

```c
/*
int main()
{
    int status;
    pid_t pid;
    if((pid=fork())>0)
    {
        wait(&status);//자식 프로세스 우선 실행 이랑 같다.
        printf("prompt>\n");//ps -e -f 기능 실행후 문자열 출력.
    }
    else if(pid==0)
    {
        execlp("ps","ps","-e","-f",0);
    }
    return 0;
}
*/
```

```
118 /*
119 //newpgm.c
120 int main(int argc,char **argv)
121 {
122     int i;
123     for(i=0;argv[i];i++)
124         printf("argv[%d]=[%s]\n",i,argv[i]);
125     return 0;
126 }
127 */
128
129 /*
130 int main()
131 {
132     int status;
133     pid_t pid;
134     if((pid=fork())>0)
135     {
136         wait(&status);
137         printf("prompt>\n");
138     }
139     else if(pid==0)
140     {
141         execl("./newpgm","newpgm","one","two",(char *)0);//내가 만든 newpgm 을 쓴다는것. newpgm,one two를 받아서 출력함,캐릭터 포인트로 널을 변환.
142     }
143     return 0;
144 }
145 */
146
```

```
169 int main()
170 {
171     system("date");//system은 내부적으로 fork 를 한후에 exec 를 한 것.
172     printf("after\n");
173     return 0;
174 }
175 */
```

```
178 /*
179 int my_system(char *cmd)
180 {
181     pid_t pid;
182     int status;
183     char *argv[]={"sh","-c",cmd,0};//shell, -c :  해당 커맨드 실행 , 입력으로 들어온 cmd= date실행
184     char *envp[]={0};
185     if((pid=fork())>0)
186         wait(&status);
187     else if(pid==0)
188
189
190         execve("/bin/sh",argv,envp);//다른 프로그램 실행 =bin/sh -c date
191
192 }
193
194 int main()
195 {
196     for(;;)
197     {
198     my_system("date");
199     sleep(1);//1초 간격으로 ...
200     }
201     printf("after\n");
202     return 0;
203 }
204 */
205
```

```
int main()
{
        signal(SIGINT,SIG_IGN);//ctrl + c 무시
        signal(SIGQUIT,SIG_IGN);//quit 도 무시
        signal(SIGKILL,SIG_IGN);//Kill 명령어는 막아놔도 사용가능하다.(최상위 신)
        pause();
        return 0;
}
```

kill -9 (프로세스) 데몬 프로세스도 죽일 수 있다.