TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

Instructor – Innova Lee(Sanghoon Lee)
gcccompil3r@gmail.com
Student – Hyungju Kim
mihaelkel@naver.com

System Programming

Implement ls –R option

-R option is for showing all files included in all lower directories. To implement it, You can use recursive call.

howard@ubuntu: ~/HomeworkBackup/22th

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>
#include <string.h>

void recursive_dir(char* dname);

int main(int argc,char* argv[]){
    recursive_dir(".");
    return 0;
}

void recursive_dir(char* dname){
    struct dirent* p;
    struct stat buf;
    DIR* dp;
    chdir(dname);
    dp = opendir(".");
    printf("\t%s :\n",dname);
    while(p = readdir(dp))
        printf("%s\n",p->d_name);
    rewinddir(dp);
    while(p = readdir(dp)){
        stat(p->d_name,&buf);
        //shorcut(2nd if) , strcmp returns 0 when it is true.
        if(S_ISDIR(buf.st_mode))
            if(strcmp(p->d_name,".") && strcmp(p->d_name,".."))
                recursive_dir(p->d_name);
    }
    chdir("..");
    closedir(dp);
}
```

fork() function

This is for creating a new process, the same as origin process, called "child process". It has return value. The value is pid for child process. If the parent has no child, return value be "0".

```
parent pid = 5763, cpid = 5764
howard@ubuntu:~/HomeworkBackup/22th$ child : pid = 5764, cpid = 0
```

Using fork function, we can prove the system can do context switching.

howard@ubuntu: ~/HomeworkBackup/22th

```c
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main(void){
    pid_t pid;
    int i;
    pid = fork();
    if(pid > 0){
        while(1){
            for(i = 0; i < 26; i++){
                printf("%c ",i + 'A');
                fflush(stdout);
            }
        }
    }
    else if(pid == 0){
        while(1){
            for(i = 0;i < 26;i++){
                printf("%c ",i + 'a');
                fflush(stdout);
            }
        }
    }
    else{
        perror("fork() ");
        exit(-1);
    }
    printf("\n");
    return 0;
}
```

System Programming

fork() function

```
 Y z Z a A b B c C d D e E f F g G h H
u v w x y z a Q b c R d S e T f U g V h
 I q J r K s L t M u N v O w P x Q y R
C j k D l E m F n G o H p I q J r K s L
 d W e X f Y g Z h A i ^C
howard@ubuntu:~/HomeworkBackup/22th$
```

You can see the processes are executing alternately. that is, OS offers "context switching".

when child process is finished, parent should receive signal. But the parent couldn't receive(by reason of sleep etc..), child become defunct state.

```
howard     3754  2166  0 Mar22 pts/18   00:00:00 ./a.out
howard     3755  3754  0 Mar22 pts/18   00:00:00 ./a.out
howard     5892  2166  0 03:12 pts/18   00:00:00 ./a.out
howard     5893  5892  0 03:12 pts/18   00:00:00 [a.out] <defunct>
howard     5895  5874  0 03:12 pts/4    00:00:00 grep --color=auto a.out
howard@ubuntu:~/HomeworkBackup/22th$
```

wait() function

it is waiting until the child process be finished. When the child process ends, it returns extract status. Below is the kill list (can check with kill –l instructor)

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Normal exit

Abnormal exit

Normal extract status has upper 8bits, and abnormal has lower 8 bits. You can see the correct value by bit operating.
Normal extract : status >> 8
Abnormal extract : status & 0x7f
->last 1 bit(128) has other option.

System Programming

Pipe communication

when implementing real-time communication system, we use non-block option. But, using multiple-process, we can make it with block.

howard@ubuntu: ~/HomeworkBackup/22th

```c
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>

int main(void){
    int fd, ret;
    char buf[1024];
    pid_t pid;
    fd = open("myfifo",O_RDWR);
    if((pid = fork()) > 0){
        for(;;){
            ret = read(0, buf, sizeof(buf));
            buf[ret] = 0;
            printf("Keyboard : %s\n",buf);
        }
    }
    else if(pid == 0){
        for(;;){
            ret = read(fd, buf, sizeof(buf));
            buf[ret] = 0;
            printf("myfifo : %s\n",buf);
        }
    }
    else{
        perror("fork()");
        exit(-1);
    }
    close(fd);
    return 0;
}
```

Family process

process has family pointer. pptr, cptr, yptr, optr, link(for priority queue).