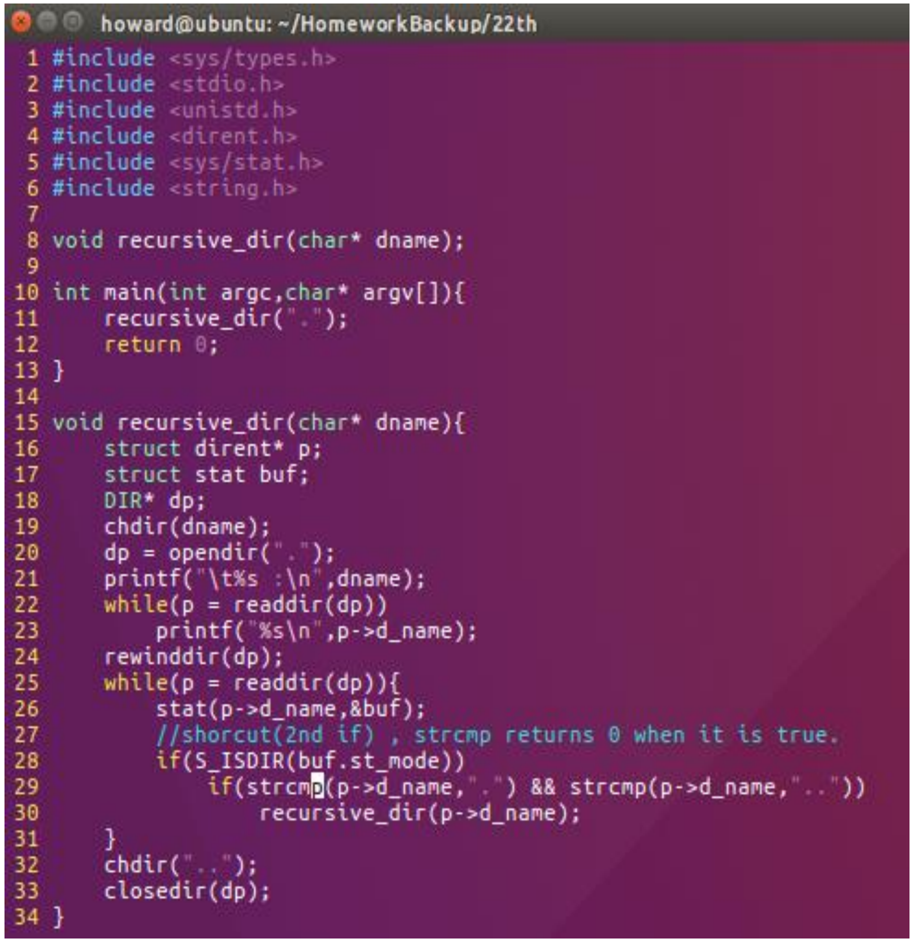
TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

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.

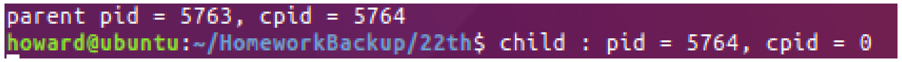


howard@ubuntu: ~/HomeworkBackup/22th

```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);
}
```

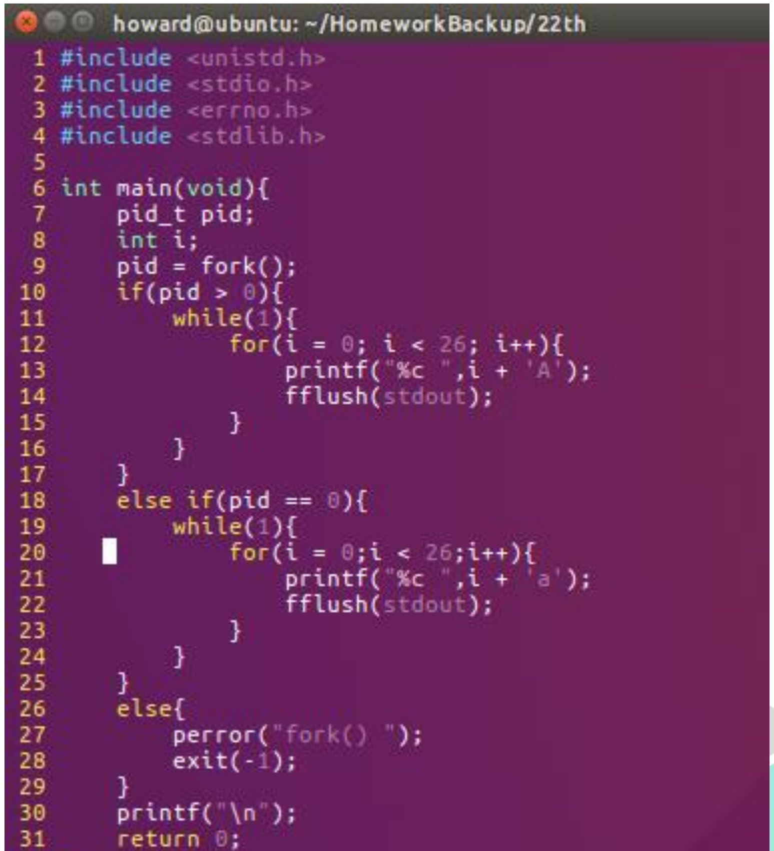
fork() function

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.

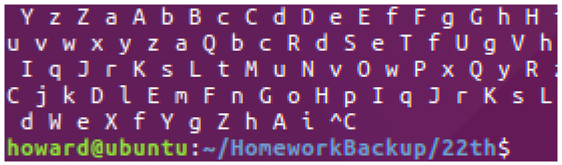


howard@ubuntu: ~/HomeworkBackup/22th

```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

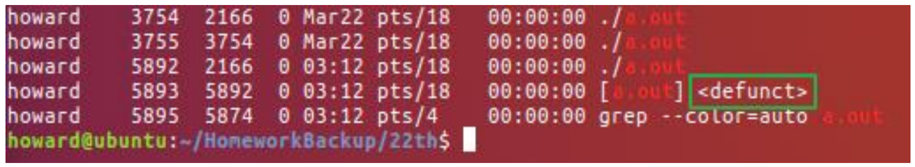
fork() function



```
 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".

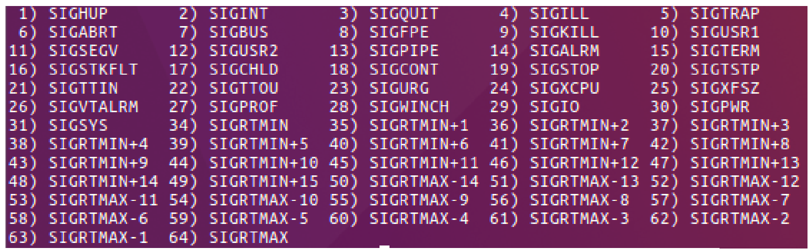
when child process is finished, parent should receive signal. But the parent couldn't receive(by reason of sleep etc..), child become defunct state.



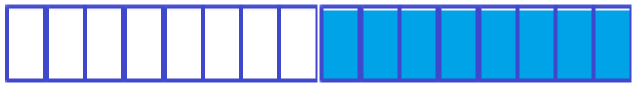
```
howard    3754  2166  0 Mar22 pts/18   00:00:00 ./a.out
howard    3755  3754  0 Mar22 pts/18   00:00:00 ./a.out
howard    5892  2166  0 03:12 pts/18   00:00:00 ./a.out
howard    5893  5892  0 03:12 pts/18   00:00:00 [a.out] <defunct>
howard    5895  5874  0 03:12 pts/4    00:00:00 grep --color=auto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

wait() function

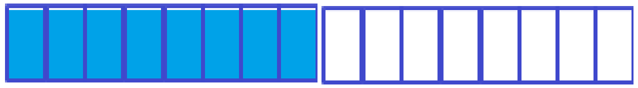
it is waiting until the child process be finished. When the child process ends, it returns extract status. Below is the kill list (can check with kill –l instructor)



```
 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



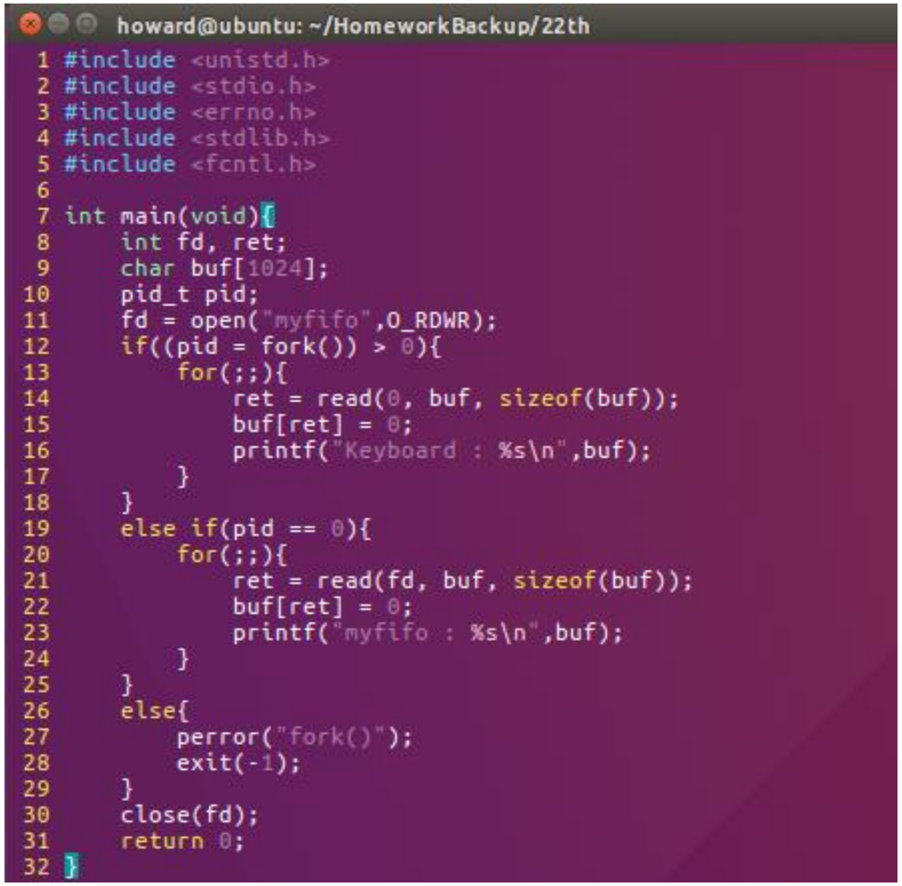
Abnormal exit

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

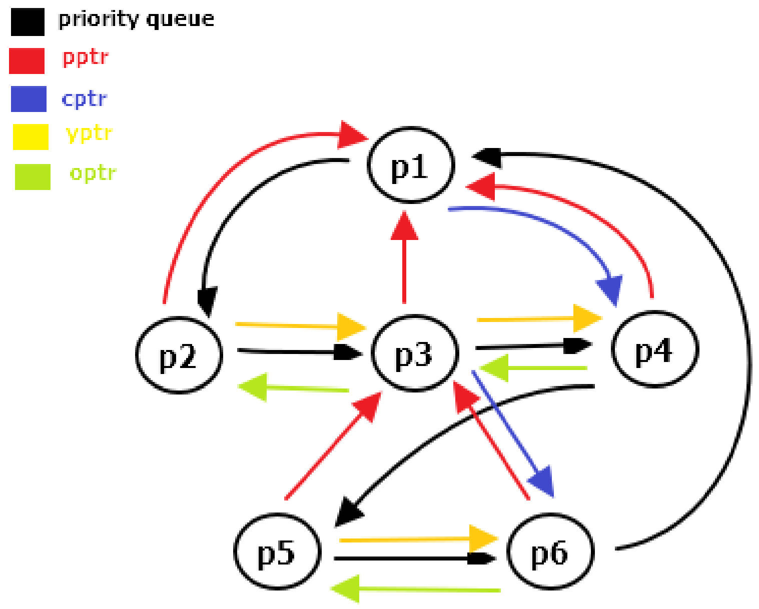
when implementing real-time communication system, we use non-block option. But, using multiple-process, we can make it with block.



```
howard@ubuntu: ~/HomeworkBackup/22th
#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).



Homework

1. a.txt has string data like below.

   apple
   banana
   peach
   mango
   watermelon
   cherry
   strawberry

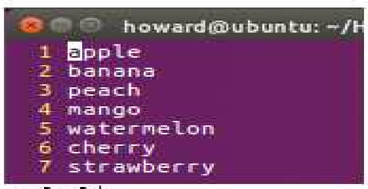
   read these data, write the sums of ascii code in each row to b.txt
   and do the same thing to .c.txt in column side.
   finally, implement ls -a function.

```c
/*
    file name : hw.c
    description :
    this is made for Homework

    execute example : ./a.out a.txt (-a)
    -a is optional.
    required file : a.txt
    apple
    banana
    peach
    mango
    watermelon
    cherry
    strawberry
    result file : b.txt .c.txt
    contact:
    e-mail : mihaelkel@naver.com
    phone  : 82 -10-9132-6404(kr)
*/

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>

void read_txt_to_ascii_raw(int argc,char** argv,int* fd);
void set_raw_str_to_ascii(char* str,int fd);
void read_txt_to_ascii_col(int argc,char** argv,int* fd);
void set_col_str_to_ascii(char* str,int fd);
void exec_ls_a(void);
int main(int argc,char** argv){
    int fd;
    read_txt_to_ascii_raw(argc,argv,&fd);
    read_txt_to_ascii_col(argc,argv,&fd);
    exec_ls_a();
    close(fd);
}
void read_txt_to_ascii_raw(int argc,char** argv,int* fd){
    int new_fd;
    char buf[1024] = {'\0',};
    int ret;
    //a.txt b.txt open
    *fd = open(argv[1], O_RDONLY, 0644);
    new_fd = open("b.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644);
    while(ret = read(*fd, buf, sizeof(buf))){
        set_raw_str_to_ascii(buf,new_fd);
    }
    close(new_fd);

}
void set_raw_str_to_ascii(char* str,int fd){
    char buf[1024]={'\0',};
    char res[1024]={'\0',};
    int i = 0;
    int ascii[1024] = {0,};
    int asc_idk = 0;

    while(str[i++]){
        if(str[i] != '\n')
            ascii[asc_idk] += str[i];
        else
            asc_idk++;
    }
    for(i=0;i<asc_idk;i++){
        if(((i+1)%10 == 1)&&(i+1 != 11))
            sprintf(buf,"%dst raw's ascii sum : %d\n",i+1,ascii[i]);
        else if(((i+1)%10 == 2)&&(i+1 != 12))
            sprintf(buf,"%dnd raw's ascii sum : %d\n",i+1,ascii[i]);
        else if(((i+1)%10 == 3)&&(i+1 != 13))
            sprintf(buf,"%drd raw's ascii sum : %d\n",i+1,ascii[i]);
        else
            sprintf(buf,"%dth raw's ascii sum : %d\n",i+1,ascii[i]);
        strcat(res,buf);
    }
    write(fd, res, strlen(res));
}
void read_txt_to_ascii_col(int argc,char** argv,int* fd){
    int new_fd;
    char buf[1024] = {'\0',};
    int ret;
    //a.txt .c.txt open
    *fd = open(argv[1], O_RDONLY, 0644);
    new_fd = open(".c.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644);
    while(ret = read(*fd, buf, sizeof(buf))){
        set_col_str_to_ascii(buf,new_fd);
    }
    close(new_fd);

}
void set_col_str_to_ascii(char* str,int fd){
    char buf[1024]={'\0',};
    char res[1024]={'\0',};
    int i = 0;
    int ascii[1024] = {0,};
    int col_idk = 0;
    int max_col;
    while(str[i++]){
        if(str[i] != '\n')
            ascii[col_idk++] += str[i];
        else{
            if(max_col < col_idk)
```

```c

                max_col = col_idk;
            col_idk = 0;
        }
    }
    strcat(res,"ascii sum(column side) : \n");
    for(i=0;i<max_col;i++){
        if(((i+1)%10 == 1)&&(i+1 != 11))
            sprintf(buf,"%dst        ",i+1);
        else if(((i+1)%10 == 2)&&(i+1 != 12))
            sprintf(buf,"%dnd        ",i+1);
        else if(((i+1)%10 == 3)&&(i+1 != 13))
            sprintf(buf,"%drd        ",i+1);
        else
            sprintf(buf,"%dth        ",i+1);
        strcat(res,buf);
    }
    strcat(res,"\n");
    for(i=0;i<max_col;i++){
        sprintf(buf,"%-8d",ascii[i]);
        strcat(res,buf);
    }
    write(fd, res, strlen(res));
}
void exec_ls_a(void){
    DIR* dp;
    int i = 0;
    struct dirent* p;
    dp = opendir(".");
    while(p = readdir(dp)){
        printf("%-16s ",p->d_name);
        if((i+1)%5 == 0)
            printf("\n");
        i++;
    }
    printf("\n");
    closedir(dp);

}

```

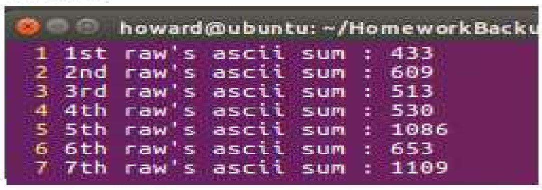
Colored by Color Scripter

a.txt



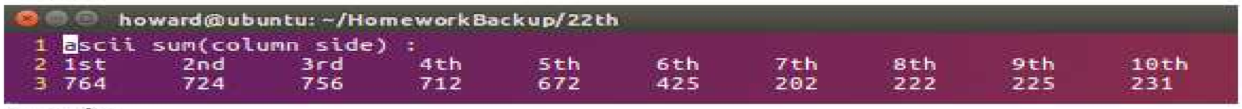
```
apple
banana
peach
mango
watermelon
cherry
strawberry
```

b.txt



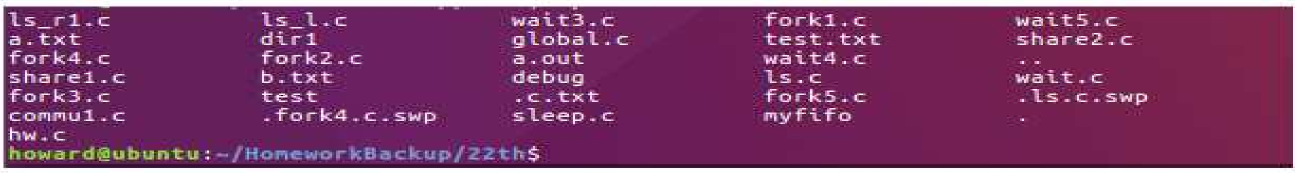
```
1st raw's ascii sum : 433
2nd raw's ascii sum : 609
3rd raw's ascii sum : 513
4th raw's ascii sum : 530
5th raw's ascii sum : 1086
6th raw's ascii sum : 653
7th raw's ascii sum : 1109
```

.c.txt



```
ascii sum(column side) :
1st     2nd     3rd     4th     5th     6th     7th     8th     9th     10th
764     724     756     712     672     425     202     222     225     231
```

result

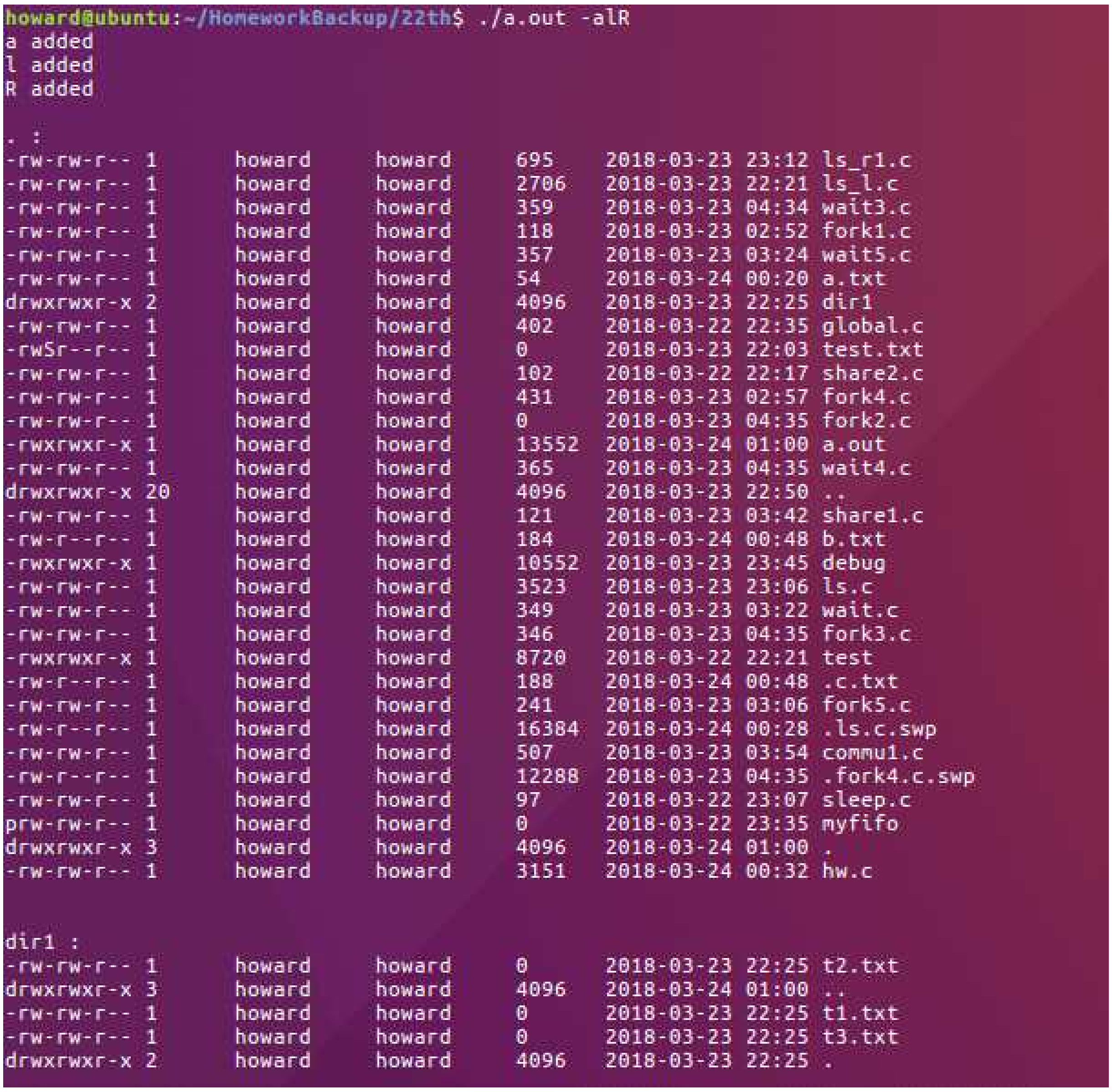


```
ls_r1.c          ls_l.c           wait3.c          fork1.c          wait5.c
a.txt            dir1             global.c         test.txt         share2.c
fork4.c          fork2.c          a.out            wait4.c          ..
share1.c         b.txt            debug            ls.c             wait.c
fork3.c          test             .c.txt           fork5.c          .ls.c.swp
commu1.c         .fork4.c.swp     sleep.c          myfifo           .
hw.c
howard@ubuntu:~/HomeworkBackup/22th$
```

2. Implement ls instructor with -a, -l, -R options

```c
/*
    file name : ls.c
    description :
    this file is for making ls instructor with 3 options, -a, -l, -R
    contact :
    e-mail : mihaelkel@naver.com
    phone : -82 10-9132-6404(kr)
*/
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>
#include <stdio.h>
#include <fcntl.h>
#include <dirent.h>
#include <unistd.h>
#include <string.h>
int read_ls_opt(int argc,char** argv,char* ins);
void ls_start(int flag,char* dname);
void adj_l_opt(struct dirent* p);
int main(int argc,char** argv){
    char* ins = "alR";
    int flag;
    char* s_dir;

    //search otehr directory
    //ex)ls ../.. -alR
    if(argv[1][0] != '-')
        s_dir = argv[1];
    else
        s_dir = ".";

    //set flag
    flag = read_ls_opt(argc,argv,ins);

    //ls start path in c_dir, with flag option.
    ls_start(flag,s_dir);


    return 0;
}
int read_ls_opt(int argc,char** argv,char* ins){
    int flag = 0b00000000;
    int cmd;
    while((cmd = getopt(argc, argv, ins)) > 0){
        switch(cmd){
            case 'a':
                flag |= 0b00000001;
                printf("%c added\n",ins[0]);
                break;
            case 'l':
                flag |= 0b00000010;
                printf("%c added\n",ins[1]);
                break;
            case 'R':
                flag |= 0b00000100;
                printf("%c added\n",ins[2]);
                break;
            }
        }
        return flag;
}
void ls_start(int flag,char* dname){
        DIR *dp;
        int i = 0;
        struct dirent* p;
        struct stat buf;
        //current directory open
        chdir(dname);
        dp = opendir(".");
        //-R option check, print directory name
        if(flag & 0b00000100)
            printf("\n%s :\n",dname);
        while(p = readdir(dp)){
            //-a option check.
            if(!(flag & 0b00000001)){
                if(p->d_name[0] == '.')
                    continue;
            }

            //-l option chkeck.
            if(flag & 0b00000010){
                adj_l_opt(p);
            }
            printf("%-16s ",p->d_name);

            //displaying 5 files, go to next line.(when -l is not added)
            //displaying 1 file , in 1 line.(when -l is addedd)
            if(!(flag & 0b00000010)){
                if((i + 1) % 5 == 0)
                    printf("\n");
                i++;
            }
            else
                printf("\n");
        }
        printf("\n");
        //-R option adjust.
        if(flag & 0b00000100){
            rewinddir(dp);
            while(p = readdir(dp)){
                stat(p->d_name,&buf);
                if(S_ISDIR(buf.st_mode))
                    if(strcmp(p->d_name,".") && strcmp(p->d_name,".."))
                        ls_start(flag,p->d_name);
            }
        }
        //close the working directory
        chdir("..");
        closedir(dp);
}
void adj_l_opt(struct dirent* p){
        struct stat buf;
        struct passwd* pw;
        struct group* gr;
        struct tm* tm;
        char ch;
```

```c

    //permission has(nrwxrwxrwx)
    char perm[11] = "----------";
    char rwx[4] = "rwx";
    char sst[4] = "sst";
    int i;
    stat(p->d_name, &buf);

    //check the current file's type
    if(S_ISDIR(buf.st_mode))
        perm[0] = 'd';
    if(S_ISREG(buf.st_mode))
        perm[0] = '-';
    if(S_ISFIFO(buf.st_mode))
        perm[0] = 'p';
    if(S_ISSOCK(buf.st_mode))
        perm[0] = 's';
    if(S_ISCHR(buf.st_mode))
        perm[0] = 'c';
    if(S_ISBLK(buf.st_mode))
        perm[0] = 'b';

    //rwx setting
    for(i=0;i<9;i++)
        if((buf.st_mode >> (8-i))&1)
            perm[i+1] = rwx[i%3];

    //sst setting
    for(i=0;i<3;i++)
        if((buf.st_mode >> (11-i))&1)
            if(perm[(i+1)*3] == '-')
                perm[(i+1)*3] = sst[i]^0x20;
            else
                perm[(i+1)*3] = sst[i];

    //print : -rwxrwxrwx 1 username groupname size YYYY-MM-DD HH:MM
    //ex    : -rw-r--r-- 1 howard   howard    4096 2018-03-24 22:10
    printf("%s ",perm);
    printf("%-6lu ",buf.st_nlink);
    pw = getpwuid(buf.st_uid);
    printf("%-10s ",pw->pw_name);
    gr = getgrgid(buf.st_gid);
    printf("%-10s ",gr->gr_name);
    printf("%-6ld ",buf.st_size);
    tm = localtime(&buf.st_mtime);
    printf("%d-%02d-%02d %02d:%02d ",
            tm->tm_year + 1900, tm->tm_mon +1,tm->tm_mday, tm->tm_hour,tm->tm_min);

}

```
Colored by Color Scripter



```
howard@ubuntu:~/HomeworkBackup/22th$ ./a.out -alR
a added
l added
R added

. :
-rw-rw-r-- 1      howard     howard     695    2018-03-23 23:12 ls_r1.c
-rw-rw-r-- 1      howard     howard     2706   2018-03-23 22:21 ls_l.c
-rw-rw-r-- 1      howard     howard     359    2018-03-23 04:34 wait3.c
-rw-rw-r-- 1      howard     howard     118    2018-03-23 02:52 fork1.c
-rw-rw-r-- 1      howard     howard     357    2018-03-23 03:24 wait5.c
-rw-rw-r-- 1      howard     howard     54     2018-03-24 00:20 a.txt
drwxrwxr-x 2      howard     howard     4096   2018-03-23 22:25 dir1
-rw-rw-r-- 1      howard     howard     402    2018-03-22 22:35 global.c
-rwSr--r-- 1      howard     howard     0      2018-03-23 22:03 test.txt
-rw-rw-r-- 1      howard     howard     102    2018-03-22 22:17 share2.c
-rw-rw-r-- 1      howard     howard     431    2018-03-23 02:57 fork4.c
-rw-rw-r-- 1      howard     howard     0      2018-03-23 04:35 fork2.c
-rwxrwxr-x 1      howard     howard     13552  2018-03-24 01:00 a.out
-rw-rw-r-- 1      howard     howard     365    2018-03-23 04:35 wait4.c
drwxrwxr-x 20     howard     howard     4096   2018-03-23 22:50 ..
-rw-rw-r-- 1      howard     howard     121    2018-03-23 03:42 share1.c
-rw-r--r-- 1      howard     howard     184    2018-03-24 00:48 b.txt
-rwxrwxr-x 1      howard     howard     10552  2018-03-23 23:45 debug
-rw-rw-r-- 1      howard     howard     3523   2018-03-23 23:06 ls.c
-rw-rw-r-- 1      howard     howard     349    2018-03-23 03:22 wait.c
-rw-rw-r-- 1      howard     howard     346    2018-03-23 04:35 fork3.c
-rwxrwxr-x 1      howard     howard     8720   2018-03-22 22:21 test
-rw-r--r-- 1      howard     howard     188    2018-03-24 00:48 .c.txt
-rw-rw-r-- 1      howard     howard     241    2018-03-23 03:06 fork5.c
-rw-r--r-- 1      howard     howard     16384  2018-03-24 00:28 .ls.c.swp
-rw-rw-r-- 1      howard     howard     507    2018-03-23 03:54 commu1.c
-rw-r--r-- 1      howard     howard     12288  2018-03-23 04:35 .fork4.c.swp
-rw-rw-r-- 1      howard     howard     97     2018-03-22 23:07 sleep.c
prw-rw-r-- 1      howard     howard     0      2018-03-22 23:35 myfifo
drwxrwxr-x 3      howard     howard     4096   2018-03-24 01:00 .
-rw-rw-r-- 1      howard     howard     3151   2018-03-24 00:32 hw.c


dir1 :
-rw-rw-r-- 1      howard     howard     0      2018-03-23 22:25 t2.txt
drwxrwxr-x 3      howard     howard     4096   2018-03-24 01:00 ..
-rw-rw-r-- 1      howard     howard     0      2018-03-23 22:25 t1.txt
-rw-rw-r-- 1      howard     howard     0      2018-03-23 22:25 t3.txt
drwxrwxr-x 2      howard     howard     4096   2018-03-23 22:25 .
```