

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

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 장성환

redmk1025@gmail.com

<p>* Paging Mechanism</p> <p>Paging 을 사용하면 어떤 부분에서 이점을 얻을 수 있을까 ?</p> <p>sticky bit 가 붙어있지 않으면 메모리가 꽉 차기 전까지 스왑을 안한다. 붙어 있는 경우는 효율적으로 스왑을 하도록 해준다.</p> <p>메모리가 엄청나게 필요하기 때문에 paging 을 사용한다. cisc rsic 차이 때문에 전력 소모 차이가 크다. (arm intel)</p> <p>mcu 는 대용량의 데이터가 입력되지 않기 때문에 페이징이 필요 없다. 즉, 대용량의 데이터를 쓰기 위해서 페이징을 쓴다는 것이다.</p> <p>mcu 는 안그래도 클럭 스피드가 느리다. 따라서 다이렉트 맵핑을 한다.</p>	
<p>* Fork</p> <p>저장소 복사 (분신술) 자기 자신을 복사한다. 하지만 복사는 하되, PID 값은 다르다.</p>	
<p>* What is Process ?</p> <p>compiler's View 입장에서 코드 데이터 힙 스택 등 가상 주소의 영역을 본다. (가상 주소(여러 프로세스)가 물리 주소에 불 연속적으로 여러군데 나뉘어 들어간다 멀티태스킹)</p> <p>운영체제 관점에서 가상 주소를 물리주소로 페이징 풀고 매치시킨다.</p> <p>하드웨어 관점에서는 그냥 물리주소가 어디에 있나 본다.</p>	<p>불연속적으로 들어가는 이유는 ? 보안</p>

<p>* VM 을 프로세서가 공유할 수 있나 ?</p> <p>ps_test (test) / ps_test2.c (a.out)</p> <p>파일 ps_test.c 를 컴파일 후에 실행 ./test &</p> <p>여기서 나오는 주소값을 ps_test2 파일에 삽입한 후에 다시 컴파일 한다.</p> <p>그리고 ps_test2 의 ./a.out 을 실행한다.</p> <p>결과는 세그멘테이션 폴트!</p>	<p>mm_struct 는 task_struct 에 있고 task 는 프로세스 마다 가지고 있다.</p> <p>b 프로세서는 a 프로세서에 대한 제어권을 가지고 있지 않다. 따라서 세그멘테이션 폴트가 발생한다.</p> <p>여러개의 프로세서가 돌아가고 있다. 하나는 모터를 제어하고 다른 것은 신호를 처리하고 있다. 하지만 VM 을 프로세서가 공유하지 못하여 신호처리 프로세서의 결과를 모터제어 프로세서에 값을 넘겨주지 못한다.</p> <p>그래서 도입된 것이 파이프, 메세지 큐, shared memory(실제 물리메모리) IPC (inter process communication)</p>
<p>* C. O. W (copy on write)</p> <p><fork5.c 확인></p> <p>fork 작업에서 메모리에 쓰기 작업이 발생할 때 복사를 한다. 한번에 복사를 하는 것이 아니다.</p>	<p>global 이 전역변수인데 갱신이 되지 않는 이유는 프로세서가 다르기 때문이다. (기계어에서 변수 선언이 먼저 되는게 아니라 값이 입력될 때, 생성이 된다.)</p>
<p>*pipe_comm.c</p> <p>프로세서 분할이 되어 블록킹이 되어도 논블록킹처럼 동작하게 된다.</p>	

process 제어의 중요한 부분 (wait.c)

mm_struct / fd / signal 세가지 가 프로세서의 가장 중요한 부분이다.

wait(&status)

자식의 상태값을 받기 위해 사용하는 함수 (자식이 어떻게 죽었는지 확인)

How to Extract Status ?

[이이이이이이이이 | | | | |] 정상종료

[| | | | | | 이이이이이이이이] 비정상 종료

따라서 256 (8 비트 시프트) 나눈값이 exit 값이 된다.

비 정상 종로의 경우 시그널이 나온다.

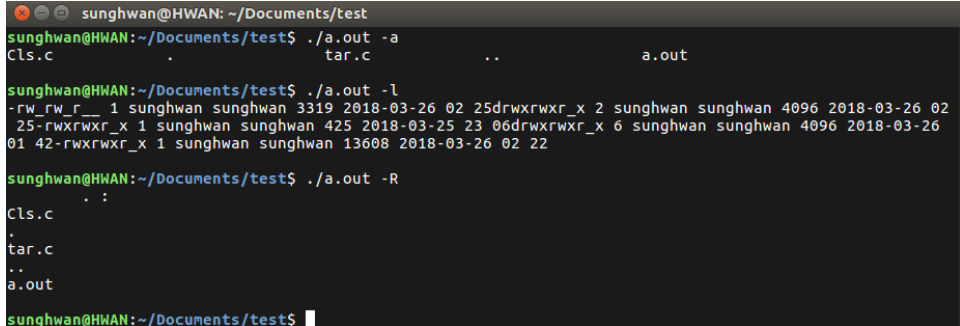
stty -a 를 하면 사용되는 시그널을 모두 보여준다.

kill -l 하면 리눅스에서 사용하든 모든 시그널을 보여준다.

*주말과제 ls -a -l -R 구현하기.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <time.h>
#include <grp.h>
#include <pwd.h>
```

// 이상한 점은 DIR 이 함수 내에서 선언이 되고 선언된 함수에서 종료가 되어야 한다...



```
sunghwan@HWAN: ~/Documents/test
sunghwan@HWAN:~/Documents/test$ ./a.out -a
cls.c      .
tar.c      ..
a.out

sunghwan@HWAN:~/Documents/test$ ./a.out -l
-rw-rw-r__ 1 sunghwan sunghwan 3319 2018-03-26 02 25drwxrwxr_x 2 sunghwan sunghwan 4096 2018-03-26 02
25-rwxrwxr_x 1 sunghwan sunghwan 425 2018-03-25 23 06drwxrwxr_x 6 sunghwan sunghwan 4096 2018-03-26
01 42-rwxrwxr_x 1 sunghwan sunghwan 13608 2018-03-26 02 22

sunghwan@HWAN:~/Documents/test$ ./a.out -R
cls.c      . :
tar.c      ..
a.out

sunghwan@HWAN:~/Documents/test$
```

```
void detail_dir(void){

    struct stat buf;
    struct dirent *p;
    struct passwd *pw;
    struct group *gr;
    struct tm *tm;
    char ch;
    char perm[11] = "_____";
    char rwx[4] = "rwx";
    char sst[4] = "sst";
    int i;
    DIR *dp;
    struct dirent *p2;
    dp = opendir(".");

    while((p2 = readdir(dp)) > 0){

        stat(p2->d_name, &buf);

        if(S_ISDIR(buf.st_mode)) //st_mode 에 해당 정보(디렉토리 인지)를 인식하여 리턴한다.
            perm[0] = 'd';
        if(S_ISREG(buf.st_mode))
            perm[0] = '-';
        if(S_ISFIFO(buf.st_mode))
            perm[0] = 'p';
        if(S_ISLNK(buf.st_mode))
            perm[0] = 'l';
        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';

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

printf("%s ",perm);
printf("%lu ",buf.st_nlink);
pw = getpwuid(buf.st_uid);
printf("%s ",pw->pw_name);
gr = getgrgid(buf.st_gid);
printf("%s ",gr->gr_name);
printf("%lu ",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);
}
printf("\n");
}

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); // pointer 를 다시 맨 처음으로 이동시킴 디렉토리의 처음 위치로
리셋.
while(p = readdir(dp)){
    stat(p->d_name, &buf);
    if(S_ISDIR(buf.st_mode))// 디렉토리 라면
        if(strcmp(p->d_name,".") && strcmp(p->d_name,"..")) // . 하고 ..
이 아니라면 1 됨
            recursive_dir(p->d_name); //short cut 기능 . 만 찾아도 바로 실행
안함
}
chdir(".."); // change dir 상위로 올라간다.
closedir(dp); //
}

void checkFlag(int flag){

    DIR *dp;
    struct dirent *dir;
    dp = opendir(".");
    int i=0;

    switch(flag){
        case 1: // -a
            while(dir = readdir(dp)){
                printf("%-16s ",dir->d_name);

```

```
        if((i+1) % 5 ==0)
            printf("\n");
        i++;
    }

    break;
case 2: // -l
    detail_dir();
    break;
case 4: // -R
    recursive_dir(".");
    break;
default:

    while(dir = readdir(dp)){
        if((dir->d_name[0] == '.')
            continue;
        printf("%-16s ",dir->d_name);
        if((i+1) % 5 ==0)
            printf("\n");
        i++;
    }

    break;
}
printf("\n");
closedir(dp);
}

int main (int argc, char **argv){
    int cmd;
    int flag;
```



```
while((cmd = getopt(argc, argv, "alR"))> 0){  
  
    switch(cmd){  
        case 'a':  
            flag |= 1<<0;  
            break;  
        case 'l':  
            flag |= 1<<1;  
            break;  
        case 'R':  
            flag |= 1<<2;  
            break;  
        default:  
            flag = -1;  
            break;  
    }  
}  
checkFlag(flag);  
  
return 0;  
}
```

* 주말과제 (파일입출력과 ls)

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

```
typedef enum _menu{
    INPUT=1,
    STATEMENT=2,
    SORT=3,
    DELETE=4,
    EXIT=5,
    TRUE=6,
    FALSE=7
}menu;
```

```
typedef struct __grades{
    char name[20];
}grade;
```

```
typedef struct _queue{
    grade *data;
    struct _queue *link;
}queue;
```

```
void enqueue(queue **head,grade *data){
    queue * tmp;
```

```
    if(*head == NULL){
        tmp = (queue*)malloc(sizeof(queue));
```

미완.

```
        tmp->link = NULL;
        tmp->data = data;
        *head = tmp;
    }
    else{
        enqueue(&(*head)->link,data);
    }
}

grade* dequeue(queue *head){
    grade *tmp;
    if(head == NULL){
        printf("no data!\n");
        return NULL;
    }
    tmp = head->data;
    head=head->link;
    return tmp;
}

void printf_queue(queue *head){
    if(head == NULL)
        return;
    printf("%10s ",head->data->name);
    printf("\n");
    printf_queue(head->link);
}

int readOneCharacter(int db,char buf[]){ //divided by " " and "\n", output
is string
```

```
int i=0;
int nread;
char tmp;
while(nread = read(db,&tmp,1)>0){
    if(tmp == '\n'){
        buf[i] = '\0';
        return TRUE;
    }
    buf[i] = tmp;
    i++;
}
buf[i]= '\0';
return FALSE;
}

void readData(queue **head,grade *person, int read_db){

    int menu=TRUE;
    char buf[1024];

    while(menu != FALSE){
        menu=readOneCharacter(read_db,buf);
        person = (grade*)malloc(sizeof(grade));
        strcpy(person->name,buf);
        enqueue(head,person);
    } // inital data load
}

void make_bt看xt(queue *head, int write_db){
    int i =0;
    int res=0;
    char tmp;
```

```
char *str[20];

if(head == NULL)
    return;
for(i=0; i<strlen(head->data->name);i++){
    tmp = (head->data->name)[i];
    res += (int)tmp;
}
sprintf(str,"%d",res);
write(write_db,str,sizeof(str));
make_bttx(head->link);
}

int main(int argc, char **argv){

    int menu;
    int read_db, write_db;
    char buf[1024];
    queue *head=NULL;
    grade *person=NULL;
    int cnt =0;

    read_db = open("a.txt",O_RDONLY,0644);
    write_db = open("b.txt",O_WRONLY,O_CREAT,O_TRUNC,0644);

    readData(&head,person,read_db);
    //printf_queue(head);
    make_bttx(head);

    close(read_db);
    close(write_db);
```

<pre>return 0; }</pre>	
----------------------------	--