2018. 3. 21 수 – 20 회차

과정 : TI, DSP, Xilinx Znq FPGA, MCU 기반의 프로그래밍 전문가 과정

Prof. 이상훈
gcccompil3r@gmail.com
Stu. 정상용
fstopdg@gmail.com

Linux 운영체제 _ 3

1. lseek
- 필요한 헤더파일 : <sys/types/h>, <unistd.h>
- 기본형태 : lseek(int filedes, off_t offset, int whence);
→ filedes : 읽기, 쓰기 포인터를 수행할 파일에 대한 file descriptor.
→ offset : whence 에서 움직일 정도를 나타내는 값(정수 – 양수, 0, 음수)
→ whence : SEEK_SET(시작점), SEEK_CUR(현재 포인터위치), SEEK_END(마지막점)
참고. SEEK_SET 일 경우, offset 에는 음수가 불가.
- lseek 를 실행할 경우, 포인터위치가 변화
- return : 시작점부터 포인터가 위치한 장소까지 byte 단위로 크기를 출력
실패할 경우는 '-1'반환

2. 압축 & 해제
→ 엄밀하게 말하자면, 압축이 아닌 '파일의 이름', '파일의 사이즈', '파일의 내용'을 res.tar 에 모음.
2.1) 압축
압축에 필요한 내용 : 파일의 이름, 파일의 사이즈, 파일의 내용

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

typedef struct
{
   char fname[20];
   int fsize;
} F_info;

int file_size(int fd)
{
   int fsize, old;
```

```
    old = lseek(fd, 0, SEEK_CUR); // 현재의 위치에  포인터를 위치시킨다. = 0
    fsize = lseek(fd, 0, SEEK_END); // 마지막 위치에 포인터를 위치시킨다. = 전체 내용의 크기
    lseek(fd, old, SEEK_SET); // 파일의 포인터를 시작점에 위치시킨다. : 포인터를 원상복구
    return fsize;
}

int main(int argc, char *argv[])
{
    int src, dst, ret;
    char buf[1024];
    F_info info;
    int i;
    dst = open(argv[argc - 1], O_WRONLY | O_CREAT | O_TRUNC, 0664); // dst : res.tar
    for(i = 0; i < argc-2; i++)
    {
      src = open(argv[i + 1], O_RDONLY); // src : a.txt, b.txt, c.txt
      strcpy(info.fname, argv[i + 1]); // info 의 name 에 "a.txt", "b.txt", "c.txt" 기입
      info.fsize = file_size(src); // info 의 size 에 각각의 내용의 크기를 기입
      write(dst, &info, sizeof(info));  // info(name, size)를 dst 에 작성(위의 두 줄에서 삽입한 내용들)
      while(ret = read(src, buf, sizeof(buf))) //  buf 에서 txt 파일의 내용을 읽음
        write(dst, buf, ret); // buf 의 내용(src)을 dst(res.tar)에 작성
      close(src);
    }
    close(dst);
    return 0;
}
```

작성된 res.tar

a.txt	f.size(a)
a.txt 의 내용	
b.txt	f.size(b)
b.txt 의 내용	
c.txt	f.size(c)
c.txt 의 내용	

2.2)해제

```
/*
압축해제
압축되서 나오는 내용
파일의 이름
파일의 내용
파일의 사이즈 - 파일의 내용을 정확히 추출하기 위한 수단
*/

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

```
#include <stdio.h>

typedef struct
{
    char fname[20];
    int fsize;
} F_info;

#define min(x, y)          (((x) < (y)) ? (x) : (y))

int main(int argc, char *argv[])
{
  int src, dst, len, ret;
  F_info info;
  char buf[1024];

  src = open(argv[1], O_RDONLY);  // src : res.tar(압축을 시킨 파일)

  while(read(src, &info, sizeof(info)))  // 한 번 read 할 경우, 첫 구조체(file 의 name, size)내용만 읽음 - sizeof(info)로 인하여
  {
    dst = open(info.fname, O_WRONLY |O_TRUNC | O_CREAT, 0644); // a.txt 를 creat
    while(info.fsize > 0)
    {
      len= min(sizeof(buf), info.fsize); // 최솟값추출
        printf("info.fsize = %d\n", info.fsize);
      ret = read(src, buf, len); // a.txt 의 크기만큼 src 를 읽음
        printf("ret = %d\n", ret);
      write(dst, buf, ret); // a.txt 의 크기만큼 읽은 내용을 dst :a.txt 에 작성
      info.fsize -= ret; // info.fsize 가 더 클 경우, ret 크기만큼 줄인 후 다시 while 문으로 들어간다.(35 번째 줄에서 sizeof(buf)가 큰경우)
    }
    close(dst);
  }
  close(src);
  return 0;
}
```

3. Keep in mind

```
#include <fcntl.h>
```

→ open 을 사용하기 위한 헤더파일

```
#include <string.h>
```

→ strlen 을 사용하기 위한 헤더파일

- man -s2 명령어 : system call 명령어 manual
- man 명령어 : 일반 명령어 manual
- cat 파일이름 : 소스코드 보여줌
-xxd 파일이름 : 압축한 파일내용 보여줌

4. dup 의 기능파악 - close(1)

```
#include <stdio.h>
#include <fcntl.h>
```

```
#include <unistd.h>

int main(void)
{
   int fd;
   fd = open("a.txt", O_WRONLY | O_CREAT | O_TRUNC, 0664);
   close(1);   //표준출력을 닫음. 즉 출력이 안됨
   dup(fd);    //종료된 것을 복사하는 역할
   printf("출력될까 ?\n");
   return 0;
}
```

→ dup : file descriptor 를 복사하는 역할(dup 앞에 close 가 있을 경우, close 한 descriptor 를 복사.)
참고. dup & close 가 아닌 dup 만 사용되는 경우
→ fd2 = dup(fd1) : fd1 의 descriptor 에서 + 1 인 fd2 의 descriptor 가 생성. descriptor 는 +1 이 되었으나 fd1 이 open 한 파일은 공동사용.

5. dup 의 기능파악 – close(0)

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

int main(void)
{
   int fd;
   char buff[1024];
   fd = open("a.txt", O_RDONLY);
   close(0);   //프로그램에서 0, 1, 2 를 한 번 닫으면 다시는 열 수 없뜸.
   dup(fd);  //close 를 한 것을 dup 가 대체받음.
   gets(buff);   //(원래는 화면에서)입력을 받는 함수, 문자열을 입력받을 수 있는 함수 -> dup(fd)의 내용을 입력받음
   printf("%s\n", buff);
   printf("출력될까 ?\n");
   return 0;
}
```

→ gets(buff) : buff 에 dup(fd)의 내용을 입력받음.
→ close 0: 표준입력, 1:표준출력, 2:표준에러 : 프로그램내에서 위 3 가지는 한 번 닫으면 다시 열수없음.

Q. a.txt 에 두 줄이상 내용이 입력될 경우, 두 줄부터는 출력이 안되는 이유(buff 의 크기도 충분히 큰 경우)

Q. xxd 의 명령어 의미

6. 원하는 위치에 char 형태로 작성(부제 : Master Boot Record)

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

```
int main(int argc, char *argv[])
{
    int i;
    char ch = 'a';
    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    lseek(fd, 512 - 1, SEEK_SET); //512 번째에 작성
    write(fd, &ch, 1);
    close(fd);
    return 0;
}
```

→ 512 번째 위치에 a 가 작성이 된다.

→ xxd 를 통하여 작성된 txt 파일을 실행.

→ 참고. 과거 플로피디스크(512byte)사용시 510byte 작성후에 끝부분에 magic number(운영체제에 관련된 정보)를 이러한 방식으로 작성 - 지금까지도 사용.

7. Blocking & Non-blocking(부제 : 채팅)

Blocking : 한 쪽에서 어떤 특정한 동작을 하기 전까지 반대쪽에서 동작이 불가.(반드시 순차적으로 작업)

Non – Blocking : 상대방의 허락을 기다릴 필요없이 편하게 동작이 가능.(Ex> 빠른 작업이 필요한 통신분야에서 주로 사용)

8. Multi – tasking