

LINUX



---

연습문제 해답




## 5장 연습문제

1. 이 장에서 작성한 cat 명령어를 수정하여 실행 인자가 없는 경우에는 표준 입력에서 읽도록 수정하라.

예제 코드 `cat3.c`를 참고하여 주십시오.

2. '\n'의 개수를 세어서 파일이 몇 줄로 구성되었는지 출력하는 명령어를 작성하라('wc-l'과 동일 기능).

예제 코드 `wc-l-syscall.c`를 참고하여 주십시오.



## 6장 연습문제

1. 탭 문자(`\t`)를 만나면 `\`와 `t`라는 두 글자로 출력하고, ‘개행’이라는 글자를 만나면 `$`과 ‘개행’으로 출력하는 `cat` 명령어를 작성하라.


예제 코드 `cat-escape.c`를 참고하여 주십시오.

2. `stdio` API를 사용하여 파일이 몇 줄로 구성되었는지 출력하는 프로그램을 작성하라. 즉, `wc -l`과 같은 기능이다. 파일 끝에 `\n`가 없는 경우에도 정상적인 결과가 나와야 한다.

예제 코드 `wc-l-stdio.c`를 참고하여 주십시오.

3. `fread()`와 `fwrite()`를 사용하여 `cat` 명령어를 작성하라.

예제 코드 `cat5.c`를 참고하여 주십시오.



## 7장 연습문제

1. 6장의 연습문제에서 만든 ‘t’나 ‘n’을 출력해 주는 기능을 cat 명령어의 옵션으로 켜고 끌 수 있게 작성하라.

예제 코드 cat4.c를 참고하여 주십시오.

2. 파일의 마지막 몇 줄을 출력하는 tail 명령어를 구현하라. 출력하는 줄의 수는 고정값으로 한다. (난이도가 있음)

예제 코드 tail.c를 참고하여 주십시오.

이 코드는 링 버퍼(ring buffer)라고 하는 데이터 구조를 사용합니다. 링 버퍼는 고정된 길이의 버퍼지만, 마지막 공간에 이어 첫 부분부터 데이터를 덮어쓸 수 있다는 점이 다릅니다.

tail 명령어의 경우, 파일을 읽으면서 언제나 마지막 10줄만 있으면 됩니다. 다시 말해, 11번째 줄을 읽었으면 첫 번째 줄은 불필요하게 됩니다. 따라서 11번째 줄의 데이터는 첫 번째 줄이 적힌 곳에 덮어써도 됩니다. 링 버퍼는 이러한 경우에 적합한 자료 구조입니다.



## 8장 연습문제

1. `grep` 명령어에 있는 `-I` 옵션과 `-v` 옵션을 이 장에서 작성한 `grep` 명령어에 추가한다.

예제 코드 `grep2.c`를 참고하여 주십시오.

2. 정규 표현식에 해당하는 줄의 전체를 출력하는 것이 아니라, 해당하는 문자열만을 출력하는 명령어 `slice`를 작성하라. 이를 위해서는 `regexexec()`의 세 번째 인자와 네 번째 인자를 사용해야 한다. 문서를 찾아보면서 진행하자.

예제 코드 `slice.c`를 참고하여 주십시오.



## 10장 연습문제

1. 인자로 전달받은 디렉터리를 재귀적으로 순회하면서 발견된 모든 파일의 경로를 표시하는 프로그램을 작성하라. 심볼릭 링크를 따라가지 않도록 주의한다.

예제 코드 `traverse.c`를 참고하여 주십시오.

2. 파일을 `open()`하고, `close()`하기 전에 그 파일을 `rename()`하면 어떤 일이 일어나는가? 또 한 `unlink()`를 하면 어떻게 되는지, 다른 파일을 `rename()`하면 어떻게 되는지 직접 실험해 보자.

`open`한 파일을 `rename()`하거나 `unlink()`해도, 이미 연결된 스트림을 사용하면 파일을 읽고 쓰는 것이 가능합니다. 이것은 유닉스 파일 시스템의 커다란 특징 중 하나입니다.

3. 인자로 지정한 경로의 디렉터리를 만들되, 상위 디렉터리가 없으면 차례대로 만드는 명령어를 작성하라(`mkdir -p`에 해당).

예제 코드 `mkpath.c`를 참고하여 주십시오.

`mkpath.c`에서는 먼저 `mkdir(2)`을 실행해 보고, 에러가 발생하면 부모 디렉터리를 만드는 전략을 채택하였습니다.

예를 들어 `mkpath a/b/c`를 실행하는 경우에는 먼저 `a/b/c`를 `mkdir`해보고, 성공하면 그것으로 종료합니다. 하지만 `mkdir`이 `ENOENT`로 실패한 경우에는 `a/b`가 존재하지 않다고 볼 수 있으므로 `make_path` 함수를 재귀&호출하여 `a/b`를 만듭니다. 이를 반복하여 디렉터리를 재귀적으로 작성합니다. 이 방법의 장점은 같은 방법으로 동작하는 `mkpath` 프로그램 여러 개가 동시에 돌아도 정상 동작한다는 점입니다.

다른 전략으로는 `stat(2)`을 사용하여 디렉터리의 존재를 확인하고, 없으면 `mkdir(2)`으로 디렉터를 만드는 방법이 있습니다. 그러나 이 방법을 사용하면 `stat(2)`로 확인한 직후에 다른 프로세스가 디렉터를 만들어버릴 수도 있으므로, `mkdir(2)`가 에러를 맞이할 가능성이 존재합니다. 하지만 `mkdir(2)`를 먼저 수행한다면, 이런 미묘한 타이밍으로 인한 에러가 발생하지 않게 됩니다.



## 11장 연습문제

1. 표준 입력의 마지막 몇 줄만 출력하는 `tail` 명령어를 작성하라. 출력할 라인 수는 실행 인자로 받아 옵션 처리하도록 한다.

예제 코드 `tail2.c`를 참고하여 주십시오. 이 명령어는 `tail.c`와 같이 링 버퍼를 사용합  
니다만, 동적으로 메모리를 확보할 필요가 있어 `malloc`을 사용한다는 점이 다릅니다.



## 12장 연습문제

1. `fork()`하면 프로세스가 사용하는 메모리가 두 배가 되는지 조사해 보라.

`fork`하는 것만으로는 메모리 사용량이 거의 증가하지 않습니다. 리눅스에서는 `fork`한 직후 두 개의 프로세스가 대부분의 논리 주소에 대한 물리 메모리를 공유합니다. 그러다 둘 중 하나의 프로세스가 메모리에 값을 쓰면, 그때서야 새로운 물리 메모리가 할당됩니다. 이 기법을 `Copy on Write`라고 합니다.

2. `fork()`와 `exec()`를 사용해서 프로그램을 기동하는 간단한 셸을 만들어라.

예제 코드 `sh1.c`를 참고하여 주십시오.

3. 문제 2에서 작성한 셸에 파이프와 리다이렉션을 구현하라(난이도가 높음).

예제 코드 `sh2.c`를 참고하여 주십시오.





## 13장 연습문제

1. SIGINT 시그널을 수신하면 메시지를 출력하고 종료하는 프로그램을 작성하라. 시그널을 기다리게 하기 위해 `pause()`라는 API를 사용할 수 있다.

예제 코드 `trap.c`를 참고하여 주십시오.

메시지를 출력한 후 `exit`하는 함수를 `sigaction(2)`으로 SIGINT의 핸들러로서 등록해 두고, `pause(2)`로 시그널을 기다립니다. 특별히 어려운 부분은 없습니다.



## 14장 연습문제

1. 셸의 pwd 명령어를 셸에서 독립된 프로그램으로 작성하는 것이 적절한가? 그 이유를 생각해 보자.

적절합니다. 왜냐하면 프로세스의 현재 작업 디렉터리는 부모 프로세스로부터 이어받기 때문에 셸의 자식 프로세스라면 셸의 현재 디렉터리를 취득할 수 있습니다.

2. 셸의 cd 명령어를 셸에서 독립된 프로그램으로 작성하는 것이 적절한가? 그 이유를 생각해 보자.

바람직하지 않습니다. 왜냐하면, 셸의 현재 작업 디렉터리는 셸 프로세스의 속성이기 때문입니다. 자기 자신 이외의 프로세스의 속성을 바꾸는 것은 기본적으로 불가능합니다. 즉, cd 명령어가 셸과 분리되어 있으면, 바꿀 수 있는 것은 cd 프로세스의 속성이기 때문에 정작 셸 프로세스의 현재 작업 디렉터리는 바뀌지 않게 됩니다.

3. 10장에서 작성한 ls 명령어를 개조하여, 파일 소유자의 사용자 이름(ID가 아닌)과 마지막 갱신 시간을 표시하도록 하라.

stat(2)를 사용하여 파일 소유자의 사용자 ID와 최종 수정 시각(time\_t)을 획득하여 사용자 ID는 getpwuid(3)를 이용해 이름으로, 시각은 ctime(3)를 이용해 문자열로 변환합니다. 어렵지는 않습니다만, 경로의 버퍼를 확보하는 것이 다소 번거롭습니다.



## 15장 연습문제

1. telnet 명령어의 사용법을 조사해서 daytime 서버에 접속해 본다. CentOS에서는 yum으로 telnet 패키지를 설치해야 한다.

‘telnet 호스트명 서비스명(혹은 포트 번호)’로 임의의 서버에 접속할 수 있습니다. 따라서 localhost의 daytime 서비스에 접속하기 위해서는 ‘telnet localhost daytime’과 같이 명령어를 실행하면 됩니다.

그러면 다음과 같이 현재 시간이 표시되고, 접속이 자동으로 끊어집니다.

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^['.
10 SEP 2017 20:57:56 JST
Connection closed by foreign host.
```

2. echo 프로토콜은 소켓에 쓴 내용을 그대로 반환해 주는 테스트용 프로토콜이다. echo 프로토콜의 클라이언트를 작성하라. echo 서버도 daytime과 같이 xinetd에 포함되어 있으므로 같은 방법으로 기동할 수 있다.

예제 코드 echoclient.c를 참고하여 주십시오.

내용은 거의 daytime 클라이언트와 같습니다. 유의할 점은 fdopen을 할 때 모드를 쓰기 가능(w+)으로 지정해야 한다는 것입니다.