설계과제 개요 : SoongSil Shell & Sed #1

Linux System Programming by Jiman Hong (jiman@ssu.ac.kr),
Sping 2017, School of CSE, Soongsil University

1. 개요

쉘(Shell)은 유닉스/리눅스 컴퓨팅 환경에서 기본적이고 중요한 부분으로, 명령 줄(Command line, cmd line)이라고도 부르며 사용자와 커널 사이의 인터페이스를 담당한다. 사용자의 명령을 해석하여 커널기능을 이용할 수 있게 해주고 자체적으로 프로그래밍 기능을 제공하여 반복적으로 수행해야 하는 명령어를 처리할 수 있다. 또, 각 사용자들의 환경을 저장하여 사용자에 맞는 환경을 제공한다.

sed 명령어는 텍스트 변환을 수행하기 위한 스트림 에디터로, ed 명령어와 grep 명령어 기능의 일부를 합친 것이다. ed명령어와 달리 기억장치 안의 버퍼를 사용하지 않고 라인들을 하나씩 읽으며, 수정하고, 출력하기 때문에 파일의 크기에 제약받지 않고 작업을 할 수 있다. 따라서 아주 큰 파일을 처리하는데 유리하다.

2. 목표

일반적인 유닉스/리눅스에 포함된 쉘(Shell) 프로그램을 구현함으로써 쉘의 원리를 이해하고, sed 명령어의 일부기능을 구현하여 시스템 프로그래밍 설계 및 응용 능력을 향상시킨다.

- 아래의 ssu_shell 및 ssu_sed 구현

명령어	내용
ssu_shell	사용자로부터 명령을 받아서 프로그램을 실행
ssu_sed	파일 혹은 디렉터리의 하위 파일에서 문자열을 검색하여 치환하는 프로그램 ssu_shell프로그램 내에 명령어로 존재

3. 팀 구성

개인별 프로젝트

4. 개발환경

가. OS : Linux

나. Tools : vi(m), gcc, gdb

5. 보고서 제출 방법

1) 제출할 파일

□ 보고서와 소스파일을 함께 압축하여 제출

- 보고서 파일 : 워드(hwp 또는 MS-Word)로 작성
- 압축파일명 : [P설계과제번호]_학번_버전.zip (예. [P1]_20132431_v1.0.zip)

2) 제출할 곳

□ http://oslab.ssu.ac.kr 접속 [Courses] ⇒ [리눅스 시스템 프로그래밍] ⇒ [과제제출]

□ 게시물 제목은 파일이름과 동일함

□ 압축된 파일을 첨부하여 과제 제출

3) 제출 기한

□ 3월 23일(수) 오후 11시 59분 59초

6. 보고서 양식

보고서는 다음과 같은 양식으로 작성

1. 과제 개요
2. 설계
3. 구현
   - 각 함수별 기능
4. 테스트 및 결과
   - 테스트 프로그램의 실행 결과를 분석
5. 소스코드(주석 포함)

7. 설계 및 구현

1) ssu_shell

가) 프로그램 설명

□ 사용자로부터 명령을 받아서 프로그램을 실행

나) 프로그램 명세

□ CMD : 실행할 프로그램 및 인자들

□ " 프롬프트" : 제작자의 학번, 공백 하나, ' $' 글자, 공백 하나를 순서대로 출력

- ex) 20170000 $

다) 세부 기능 및 기능별 요구 조건

□ 입력 요구조건

- CMD에 사용자가 원하는 기능의 프로그램의 이름과 인자를 입력
- Ctrl + C(프로그램 종료) 입력될 때 까지 명령을 입력받아 실행
- CMD에 프로그램의 이름과 인자를 합쳐 최대 10개까지 입력받을 수 있음
  - ✓ 10개를 초과 시 에러 출력
- 아무 입력 없이 엔터를 치면 프롬프트 다시 출력

□ 출력 요구조건

- 프롬프트 출력, 즉 학번, 공백,' $' 글자 및 공백 한 글자를 출력 후 사용자의 명령을 기다림
  - ✓ ssu_shell의 실행 후, 사용자의 명령을 실행 후 등 ssu_shell에서 사용자의 명령을 대기 중에는 프롬프트를 출력
- 명령어 입력 후 엔터를 치면, parsing 후 입력된 프로그램을 실행
  - ✓ 목표에 명시된 프로그램 중 ssu_shell 이외의 명령어(ssu_sed)는 ssh_shell 프로그램 내에 개별 함수로 구현 후, 해당 명령어를 ssu_shell에서 실행하면 ssu_shell 프로세스 안에서 함수로 실행. 목표에 명시된 프로그램(ssu_sed)의 실행은 system 함수를 사용하지 않음.
  - ✓ 목표에 명시되지 않은 명령어는 system() 함수로 실행
- 명령어 수행이 끝나면 명령어가 동작한 시간을 출력
  - ✓ gettimeofday를 이용하여 밀리초 단위로 " 초.밀리초" 형식으로 소수점 셋째자리까지 출력

□ 제한조건

- ssu_sed 실행시 모든 인자를 parsing하여 ssu_sed를 실행하는 함수에 넘겨줌

라) 실행 예

```
20170000 $ ls
Makefile  include  src  ssu_sed  test

time : 0.005
20170000 $ history

time : 0.000
```

2) ssu_sed

가) 프로그램 설명

□ 사용자가 지정한 파일/디렉토리의 하위파일들에서 문자열을 찾아 지정된 문자열로 치환하는 프로그램

나) 프로그램 명세

□ ssu_history [TARGET] [SRC_STR] [DEST_STR] [OPTION]

□ [TARGET]

- 치환을 실행할 대상 파일이나 디렉토리

□ [SRC_STR]

- 치환 전 문자열

□ [DEST_STR]

- 치환 후 문자열

□ [OPTION]

- -b : 이 옵션이 설정되면 문자열을 검색할 때 공백을 무시함. 단, SRC_STR의 공백을 무시하진 않음
  - ✓ ex1) ABC 치환 요청시 A BC도 치환됨
  - ✓ ex2) A BC 치환 요청시 ABC는 치환되지 않음
- -u [USERNAME] : 해당 파일의 소유자가 USERNAME과 같을 경우, 해당 파일 내의 문자들을 치환함
- -i [STRING] : STRING을 파일 내용으로 포함하는 파일인 경우, 해당 파일 내의 문자들을 치환함
- -e [STRING] : STRING을 파일 내용으로 포함하지 않는 파일인 경우, 해당 파일 내의 문자들을 치환함
- -d [DEPTH] : DEPTH에 해당하는 깊이만큼만 디렉터리 하위폴더를 탐색하고, 해당 옵션이 없는 경우 최대깊이까지 탐색함.
- -p : 치환된 문자열들의 정보 출력(파일명, 라인 등)
- -s : 대소문자 구분 안함. 접미사로 문자열을 받는 다른 옵션과도 함께 사용 가능(ex :-es, -is)
- -P [PATHNAME] : 결과를 원본파일에 덮어쓰지 않고 PATHNAME에 해당하는 경로에 저장

다) 세부 기능 및 기능별 요구사항

□ 입력 요구조건

- 모든 옵션을 동시에 쓸 수 있음 (단, parsing 후 인자의 개수가 10개를 초과할 수 없음)
  - ✓ ex) ssu_sed -b -u oslab -d 5
- 같은 옵션을 중복으로 사용할 수 없음

□ 출력 요구조건

- 인자로 받은 모든 문자열을 출력
- 변환될 파일의 절대경로를 출력하고 해당 파일의 변환 여부 또한 출력
- -p 옵션이 있는 경우, 변환된 모든 문자열의 경로명과 줄번호를 출력
  - ✓ 파일의 변환 여부도 출력해야함
- 알 수 없는 옵션이 입력되면 사용법 출력

□ 제한조건

- string.h 헤더파일 사용 금지
- printf계열, scanf계열 함수 사용 금지 (sprintf, fprintf, sscanf 등, 단, read, write, fgets 등의 함수는 사용 가능)

라) 실행 예

```
20170000 $ ssu_sed test OSLAB Soongsil\ OSLAB -u oslab
arg0 : ssu_sed
arg1 : test
arg2 : OSLAB
arg3 : Soongsil OSLAB
arg4 : -u
arg5 : oslab
/home/oslab/test/testdir1/testfile1 : success
/home/oslab/test/testdir1/testfile2 : success
/home/oslab/test/testdir2/testfile1 : failed
/home/oslab/test/testfile1 : success
/home/oslab/test/testfile2 : success

time : 0.076
```

```
20170000 $ ssu_sed test OSLAB Soongsil\ OSLAB -p -b -i EOSLAB
arg0 : ssu_sed
arg1 : test
arg2 : OSLAB
arg3 : Soongsil OSLAB
arg4 : -p
arg5 : -b
arg6 : -i
arg7 : EOSLAB
/home/oslab/test/testdir1/testfile1 : 1
/home/oslab/test/testdir1/testfile1 : 21
/home/oslab/test/testdir1/testfile1 : 87
/home/oslab/test/testdir1/testfile1 : 136
/home/oslab/test/testdir1/testfile1 : success
/home/oslab/test/testfile1 : 1
/home/oslab/test/testfile1 : 3
/home/oslab/test/testfile1 : success
/home/oslab/test/testfile2 : 4
/home/oslab/test/testfile2 : 13
/home/oslab/test/testfile2 : success

time : 0.083
```

<ssu_sed 실행결과 예시>

/home/oslab/test/testfile1 치환 전

```
OSLAB
Hello, we are the best computer science lab in korea.
OSLAB was established in 1992.
```

/home/oslab/test/testfile1 치환 후

```
Soongsil OSLAB
Hello, we are the best computer science lab in korea.
Soongsil OSLAB was established in 1992.
```

<ssu_sed 실행 후 파일 변화>

8. 설계 구성 요소

1) 목표 설정

□ 주어진 요구 조건을 이해하고 명확한 설계 목표를 설정한다.

□ 설계의 목표 및 요구조건을 문서화한다.

2) 분석

□ 목표 설정에서 명시한 요구 조건을 분석하고 해결을 위한 기본 전략을 수립한다.

□ 문제 해결을 위한 배경 지식을 이론 강의에서 듣고 개별적으로 학생들이 다양한 경로를 통해 자료를 찾아 분석한다.

3) 합성 (구조 설계)

□ 분석 결과를 토대로 적절한 구조를 도출한다.

□ 도출된 구조를 토대로 모듈을 작성한다.

4) 제작 (구현)

□ 각 모듈의 입출력을 명세한다.

□ 구조와 모듈을 C 언어로 구현하고 이를 컴파일하여 실행 파일을 만든다.

5) 시험 및 평가(성능 평가)

□ 모듈의 입출력 명세에 따라, 다양한 입력 데이타를 작성하고 모듈을 테스트한다.

□ 작성된 실행 파일이 안정적으로 수행되는지 다양하게 테스트하고 이를 평가한다.

6) 결과 도출

□ 안정적으로 수행되는 최종 결과(Output)를 캡쳐하여 최종 결과물은 보고서에 반영한다.

9. 평가 도구

1) 설계 보고서 평가

2) 실행 평가

10. 평가 준거(방법)

1) 평가 도구 1)에 대한 평가 준거

□ 소스 코드 분석 및 새로운 모듈의 설계가 제대로 이루어졌는가?

- 설계 요구 사항을 제대로 분석하였는가?
- 설계의 제약 조건을 제대로 반영하였는가?
- 설계 방법이 적절한가?

□ 문서화

- 소스 코드에 주석을 제대로 달았는가?
- 설계 보고서가 잘 조직화되고 잘 쓰여졌는가?

2) 평가 도구 2)에 대한 평가 준거

□ 요구조건에 따라 올바르게 수행되는가?

11. 기타

1) 보고서 제출 마감은 제출일 자정까지

2) 최소로 구현해야하는 옵션들 미구현 시 과제 제출 불인정

가) ssu_shell 모든 기능 구현

나) ssu_sed 1, 2기능 구현

3) 지연 제출 시 감점

□ 1일 지연 시 마다 30% 감점

□ 3일 지연 후부터는 미제출 처리

4) 압축 오류, 파일 누락

□ 50% 감점 처리 (추후 확인)

5) copy 발견시

□ F 처리

> ssu_shell 30

1. 쉘이 실행되고 system()을 이용해 명령어 실행 5
2. ssu_sed 명령어 실행 5
3. parsing 알고리즘을 사용해 ssu_sed에 인자를 정상적으로 넘김 20

> ssu_sed 70

1. 넘겨받은 인자를 모두 출력 5
2. 디렉터리 하위항목을 정상적으로 탐색 10
3. 문자열을 정상적으로 치환 25
4. -b, -u옵션 구현 총 5점
5. -i, -e 옵션 구현 총 5점
6. 이외의 옵션은 각 5점 (-s옵션의 경우, -es와 -is를 모두 구현해야 인정)