오픈소스sw개론

학교:조선대학교
학과:컴퓨터공학과
교수님:전찬준교수님
학번:20185010
이름:강세희

getopt는 명령어를 argv에 있는 명령어를 계속적으로 parsing하는데, 정상적으로 파싱이 되면 optstring에서 지정한 문자열이 반환되고 파싱이 전부되었다면 최종적으로 -1을 반환합니다.

쉘 에서 명령을 실행할 때 옵션을 사용하는데요. 스크립트 파일이나 함수를 실행할 때도 동일하게 옵션을 사용할 수 있습니다. 사용된 옵션은 다른 인수들과 마찬가지로 $1, $2, ... positional parameters 형태로 전달되므로 스크립트 내에서 직접 옵션을 해석해서 사용해야 됩니다. 이때 옵션 해석 작업을 쉽게 도와주는 명령이 getopts 입니다.

예제

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

void showHelp(const char* thisName){
        printf("usage : %s [OPTION] [MESSAGE]\n"
                        " -h                help\n"
                        " -w [MESSAGE]   write file\n"
                " -r                    read file\n"
                        , thisName);
}

void writeFile(const char* message){
        int fd = open("getopt_file.txt",O_RDWR|O_CREAT,0777);
        write(fd, message, strlen(message));
        close(fd);
}

void readFile(){
```

```
        char buf[32]={0,};
        int n;
        int fd = open("getopt_file.txt",O_RDWR|O_CREAT,0777);
        while((n = read(fd, buf, sizeof(buf))) > 0)
                printf("%s",buf);

        close(fd);
}
int main(int argc, char* argv[]){
                char opt;
                opterr = 0;

                if(argc < 2){
                        showHelp(argv[0]);
                        exit(0);
                }

                while((opt = getopt(argc, argv, "hw:r")) != -1){
                        printf("opt:%c, optopt:%d\n", opt, optopt);
                        switch(opt){
                        case 'h':
                                showHelp(argv[0]);
                                break;
                        case 'w':
                                writeFile(optarg);
                                break;

                        case 'r':
                                readFile();
                                break;

                        }
                }
}
```

기본적인 기능은 ed에서 따 왔으며, 이 기능들은 모두 sed에 적용이 된다. 다만 ed는 대화형 편집기이며,
sed는 스트리밍 편집기이다. 대화형 편집기와 스트리밍 편집기의 차이점은 대화형 편집기는 입력 및 출력이
하나로 이루어지며, 스 트리밍 편집기는 하나의 입력이 하나의 출력을 낸다는 것이다.
\n개행문자로 사용하는 스트리밍 에디터이다.

찾기(search), 출력(print),
sed -n '/abd/p' list.txt : list.txt 파일을 한줄씩 읽으면서(-n : 읽은 것을 출력하지 않음) abd 문자를 찾으면 그 줄을 출력(p)한다.

치환(substitute),
sed 's/addrass/address/' list.txt : addrass를 address로 바꾼다. 단, 원본 파일을 바꾸지 않고 출력을 바꿔서 한다.
sed 's/addrass/address/' list.txt > list2.txt
sed 's/\t/\ /' list.txt : 탭문자를 엔터로 변환
sed 's/□□*/□/' list.txt : (*표시: □ 는 공백 문자를 표시한다.) 위의 구문은 한개이상의 공백문자열을 하나의 공백으로 바꾼다.

추가(insert)
scriptfile - s/
삭제(delete)
sed '/TD/d' 1.html : TD 문자가 포함된 줄을 삭제하여 출력한다.
sed '/Src/!d' 1.html : Src 문자가 있는 줄만 지우지 않는다.
sed '1,2d' 1.html : 처음 1줄, 2줄을 지운다.
sed '/^$/d 1.html : 공백라인을 삭제하는 명령이다
파 일 이름만을 뽑아내는 정규식
s/^.*\/\([a-zA-Z0-9.]*\)".*$/\1/ : ^는 라인의 맨 처음, .* 아무문자열, \(, \)은 정규표현식을 그룹화, $ 는 라인의 맨 끝.
(s;^.*\/\([a-zA-Z0-9.]*\)".*$;\1;) \1는 그룹화된 첫번째 요소를 말한다.
[a-zA-Z0-9.] 는 알파벳과 숫자 및 .(콤마)를 표현하는 문자(character)를 말한다.

즉 GF02.jpg와 같은 문자열을 첫번째 그룹화하고 난 다음 라인 전체를 그룹화된 내용으로 바꾸는 것이다.

/g : global을 의미 한줄에 대상문자가 여러개일 때도 처리하도록 한다.

who | sed -e 's; .*$;;' : 각 라인의 첫 번째 공백에서부터 마지막까지 삭제하라.

who | sed -e 's;^.* ;;' : 각 라인의 처음부터 맨 마지막 공백까지 삭제하라.

who | sed -e 's;^.*:;;' : 각 라인의 처음부터 : 문자가 있는 곳(:문자포함)까지 삭제하라.

-n 옵션
sed는 항상 표준 출력에서 입력 받은 각 라인을 나타낸다는 것을 알아냈다. 그러나 때때로 한 파일로부터 몇 개의 라인들을 추출해 내기 위해 sed를 사용하기를 원할 때도 있다. 이러한 경우에 -n옵션을 사용한다. 이 옵션은 사용자가 만약 해야 할 일을 정확히 말해 주지 않는다면 임의의 라인을 프린트하는 것을 원하지 않는다고 sed에게 말한다. 따라서 p명령이 같이 쓰인다. 라인 번호와 라인 번호의 범위를 나타냄으로써 sed를 사용하여 텍스트의 라인들을 선택적으로 프린트할 수 있게 한다. 다음에서 볼 수 있는 바와 같이, 한 파일로부터 첫 번째 두 라인이 프린트되었다.

$ sed -n '1,2p' intro Just print the first 2 lines from intro file.

만약 라인 번호 대신에 슬래시로 에워 싸인 문자열과 함께 p명령이 쓰인다면 sed는 이들 문자들이 포함하고 있는 표준 입력을 통해서 라인들을 프린트하게 된다. 따라서 하나의 파일로부터 처음의 두 라인을 프린트하기 위하여 다음과 같이 사용될 수 있다.

$ sed -n '/UNIX/p' intro Just print lines containing UNIX
sed '5d' : 라인 5를 삭제
sed '/[Tt]est/d' : Test 또는 test를 포함하는 모든 라인들을 삭제
sed -n '20,25p' text : text로부터 20에서 25까지의 라인들만 프린트

sed '1,10s/unix/UNIX/g' intro : intro의 처음 10개의 라인들의 unix를 UNIX로 변경
sed '/jan/s/-1/-5' : jan을 포함하는 모든 라인들 위의 첫 번째 -1을 -5로 변경
sed 's/...//' data : 각 data라인으로부터 처음 세 개의 문자들을 삭제
sed 's/...$//' data : 각 데이터 라인으로부터 마지막 3문자들을 삭제
sed -n '1' text : 비 프린트 문자들을 \nn으로 (여기서 nn은 그 문자의 8진수 값임),
그 리고 탭 문자들을 > 로 나타내는 각 텍스트로부터의 모든 라인들을 프린트

awk 명령어

awk '/west/' datafile : west 라는 글이 있는 줄 출력
awk '/^north/' datafile : north로 시작하는 줄 출력
awk '/^(no | so)/' datafile : no 또는 so 로 시작하는 줄 출력
awk '{ print $3, $2 }' datafile : datafile 리스트의 세 번째 와 두 번째 필드를 스페이스로 띄어서 출력
awk '{ print $3 $2 }' datafile : datafile 리스트의 세 번째 와 두 번째 필드를 그냥 붙여서 출력
awk '{ print "Number of fields : " NF} ' datafile : datafile의 각 줄마다의 필드수를 리턴한다.
awk '$5 ~ /\.[7-9]+/' datafile : 다섯 번째 필드가 마침표 다음엣 7과 9사이 숫자가 하나 이상 나오는 레코드 출력
awk '$2 !~ /E/ { print $1, $2 }' datafile : 두 번째 필드에 E 패턴이 없는 레코드의 첫 번째와 두 번째 필드 출력
awk '$3 ~ /^Joel/{ print $3 " is a nice guy."} ' datafile : 세 번째 필드가 Joel로 시작하면 " is a nice guy"와 함께 출력
awk '$8 ~ /[0-9][0-9]$/ { print $8 }' datafile : 여덟 번째 필드가 두 개의 숫자이면 그 필드가 출력
awk '$4 ~ /Chin$/ { print "The price is $" $8 "." }' datafile : 네 번째 필드가 Chine으로 끝나면 "The price is $" 8번 필드 및 마침표가 출력
awk -F: '{ print $1 } ' datafile : -F 옵션은 입력 필드를 ':'로 구별.
awk -F"[:]" '{ print $1, $2 } ' datafile : 입력 필드로 스페이스와 ':'를 필드 구별자로 사용

awk -f awk_script.file datafile : -f 옵션은 awk 스크립트 파일 사용할 때 씀.

awk '$7 == 5' datafile : 7번 필드가 5와 같다면 출력
awk '$2 == "CT" { print $1, $2 }' datafile : 2번 필드가 "CT" 문자와 같으면 1, 2 번 필드 출력
awk '$7 < 5 { print $4, $7}' datafile : 7번 필드가 5보다 작다면 4번, 7번 필드 출력
awk '$6 > .9 { print $1, $6}' datafile : 6번 필드가 .9 보다 크다면 1번, 6번 출력
awk '$8 > 10 && $8 < 17 ' datafile
awk '$2 == "NW" || $1 ~ /south/ { print $1, $2 }' datafile

https://github.com/sehui99/oss.git