시스템 프로그래밍

Linux system programming

명령어 옵션

KYUNGSUNG UNIVERSITY SINCE 1955

getopt

```
int getopt(int argc, char * const argv[], const char *optstring);
```

getopt() 는 입력라인의 인자(arguments)를 분석한다. 프로그램 실행에 의해 main()함 수에서 넘겨진 argc와 argv는 인자의 수와 배열을 나타낸다. '-' 또는 '--'를 정확히 구분을 하지않지만 '-'으로 시작되는 argv의 요소가 옵션 요소(option element)가 된다. '-'으로 시작하여 뒤에 있는 문자는 옵션문자(option characters)가 된다. getopt() 를 반복적으로 호출하게 되면 각각의 옵션인자(option element)에서 각각의 옵션 문자(option characters)들이 성공적으로 반환된다.
만일 getopt()가 또 다른 옵션 문자가 있음을 발견하게 된다면, 이 문자를 반환하고, 외부 변수 optind와 정적 변수 nextchar를 업데이트함으로서, getopt() 함수가 다시 호출되었을때 현재 검색된 다음 인자(argv)또는 다음 옵션부터 검색할수 있도록 있도록 한다.
만일 더이상의 옵션 문자가 없다면, getopt()는 -1 을 반환하며, optind는 argv들중에 첫번째 argv-요소를 가르킴으로서 더이상 옵션이 존재하지 않음을 알린다. (주:argv의 첫번째는 프로그램 자신이며, 옵션이 아니죠.)
optstring optstring은 규칙에 맞는 옵션 문자들이 포함되어 있는 문자열을 나타낸다. 만일 이러한 문자뒤에 콜론(:)이 있다면, 해당 옵션이 인자를 요구한다는 것을 의미한다. 따라서, getopt는 현재 argv-요소에 뒤따르는 텍스트 부분으로 포인터를 이동시키거나, .IR optarg내의 다음 argv요소에 있는 텍스트 부분으로 포인터를 이동시킨다. 두개의 콜론(::)은 옵션이 추가적인 인자를 요구함을 나타낸다; 만일 현재 argv-요소에 텍스트가 존재한다면, optarg으로 반환되며, 그렇지 않다면 optarg는 '0'의 값을 지니게 된다. 이것은 GNU 확장이다. optstring이 ';'(semicolon)을 수반하는 W를 포함한다면, -W foo는 long option --foo로써 다루어진다. (kim: foo가 하나의 arg로 다루어진다는 건지 테스트 필요) (-W option은 구현 확장들에 대해 POSIX.2에 의해 예약되어있다.) 이 동작은 GNU extension이며, glibc 2 이전의 library에서 이용가능하지 않다.

기본적으로, getopt()는 argv를 읽음으로써 argv의 내용을 재배치하여 결과적으로 모든 non-option들을 뒤에 놓는다. 두개의 다른 mode들이 또한 구현되어있다. optstring의 첫번째 character가 '+' 이거나 환경변수 POSIXLY_CORRECT가 설정되어있다면, option processing은 non-option argument와 마주치면서 곧바로 멈춘다. optionstring의 첫번째 character가 '-'이라면, 각 non-option argv-element는 마치 그것이 chracter code 1인 option의 argument로써 다루어진다. (이것은 options들과 다른 argv-element들이 어떤 순서 혹은 두가지 처리에 대해 유념하는 것이 있을 거라는 것을 예상하고 쓰여진 프로그램에 의해 사용된다. special argument "--"는 scanning mode에 관계없이 option-scanning의 마지막을 강요한다.
getopt()가 option character를 알아차리지 못한다면, stderr로 error message를 출력하고, optopt안에 character를 저장한며 '?'를 return한다. 호출하는 program은 opterr를 0으로 설정함으로서 error message를 방지한다.
getopt()가 optstring에 포함되지 않은 argv안에 option character를 찾거나 missing option argument를 감지한다면, '?'를 return하고 external variable optopt를 실제 option character로 설정한다. optstring의 (위에 서술된 어떤 추가적인 '+' 또는 '-'를 수반하는) 첫번째 character가 ':'(colon)이라면 getopt는 missing option argument를 가리키는 '?' 대신에 ':'를 리턴한다. error가 감지되었다면, optstring의 첫번째 character는 colon(':')이 아니고 external variable opterr는 0이 아니며,(기본적으로) error message를 출력한다.

getopt_long 과 getopt_long_only

getopt_long()함수는 2개의 dash로 시작되는 long options을 허용하는것을 제외하고는 getopt()와 같이 동작한다. (프로그램이 long option만이 허용된다면, optstring은 empty string ("")로써 명시되어질 수 있다, NULL이 아님) Long option name은 어떤 정의된 option에 대해 정확히 대응되거나 유일한 단축이라면 생략될 수 있다. long option은 arg=param 또는 arg param 형태의 parameter를 취할 수 있다.
longopts는 다음의 <getopt.h>에 선언되어진 struct option의 배열의 첫번째 요소에 대한 pointer이다.

```
struct option {
    const char *name;
    int         has_arg;
    int        *flag;
    int         val;
};
```

각 field들의 의미는:
name
long option의 이름
has_arg
option이 argument를 취하지 않는다면 no_argument(또는 0);

option이 하나의 argument를 취한다면 required_argument(또는 1); 또는
option이 추가적인 argument를 취한다면 optional_argument(또는 2).
flag
어떻게 결과가 long option에 대해 return되는지 명시한다. flag가 NULL이면, getopt_long()는 val을 return한다. (예를 들어, 호출하는 program은 동등한 short option character로 val설정 할 것이다.) 다른 경우라면, getopt_long()는 0을 return하고 flag는 option이 발견되면 val로 설정한 variable을 가리킨다. 그러나 option이 발견되지 않으면 바뀌지 않게 내비둔다.
val
return되는 값이거나 flag에 의해 가리키는 variable로 로드한다.
array의 마지막 element는 0으로 채워져야 한다.
longindex가 NULL이 아니라면 longopts에 상대적인 long option의 index로 설정된 variable을 가리킨다.

리턴값

option이 성공적으로 찾아지면, getopt()는 option character를 return한다. 모든 command-line option들이 파싱 되어진다면, getopt()은 -1을 return한다. getopt() 이 optstring안에 없는 option character와 마주치면, '?'이 return된다. getopt() 이 missing argument을 갖는 option에 마주치면, return value는 optstring내의 첫번 째 character에 의존한다.: ':'이라면, ':'이 return된다; 다른 경우 '?'이 return된다.
getopt_long()과 getopt_long_only() 또한 short option을 알아볼 때 option character가 return된다. long option에 대해, flag가 NULL이라면 val을 retun하 고, 다른경우는 0이다. Error와 -1 return 은 getopt()에 관해서 같다, 모호하게 대응하거나 관계 없 는 parameter에 대해 '?'가 더해진다.

int getopt(int argc, char * const argv[], const char *optstring);

이 함수의 파라미터는 간단한다.

argc, argv : main() 함수가 받은 파라미터를 그대로 전달한다.
optstring : 파싱해야 할 파라미터를 쓴다. 옵션이 별도의 파라미터를 받는 경우 콜론을 함께 쓴다.
예를 들어 -h, -v, -f filename을 받는 세 가지 옵션이 있다고 하면 옵션스트링은 "hvf:"가 된다. 각각의 옵션을 파싱해내기 위해서는 getopt()함수가 0을 리턴할 때까지 계속해서 반복하면 된다.

그외 전역변수
이 함수와 관련된 전역 변수에는 다음과 같은 것들이 있다.

optarg : 옵션 뒤에 별도의 파라미터 값이 오는 경우, 이를 파싱한 결과 파라미터 값은 optarg에 문자열로 저장된다.
optind : 다음번 처리될 옵션의 인덱스이다. 만약 파싱한 옵션이후에 추가적인 파라미터를 받는다면 (예를 들어 입력 파일 이름 같이) 이 값을 활용할 수 있다. getopt()함수는 한 번 호출될 때마다 이 값을 업데이트한다.
opterr : 옵션에 문제가 있을 때, 이 값은 0이 아닌 값이되며, getopt()함수가 메시지를 표시하게 된다.
optopt : 알 수 없는 옵션을 만났을 때 해당 옵션이 여기에 들어간다. (이 때 getopt의 리턴값은 '?'가 된다.)

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

void help();
void version();
int main(int argc, char **argv)
{
    int opt;
    int port_num;
    int fp;
    int opt_ok;
    char file_name[16];
    char buf[256];

    while((opt = getopt(argc, argv, "hvf:")) != -1)
    {
        switch(opt)
        {
            case 'h':
                help();
                break;
            case 'v':
                version();
                break;
            case 'f':
                memcpy(file_name, optarg, 16);
                opt_ok = 1;
                break;
        }
    }
```

```
 if (opt_ok != 1)
    {
        help();
        exit(0);
    }

    if (access(file_name, R_OK) != 0)
    {
        printf("파일이 존재하지 않습니다");
        exit(0);
    }
    if((fp = open(file_name, O_RDONLY)) == -1)
    {
        printf("file open error");
        exit(0);
    }

    memset(buf, '0', 256);
    while(read(fp, buf, 256) > 0)
    {
        printf("%s", buf);
        memset(buf, '0', 256);
    }
}
```

```
void help()
{
    printf("Usage: ./testopt [OPTION] [FILE]"
           "   -h              도움말"
           "   -f [FILE]       파일출력"
           "   -v              버전출력");
    exit(0);
}

void version()
{
    printf("Version : 1.01");
    exit(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>

void help(char *argv)
{
    printf("Usage : %s options₩n", argv);
    printf("--debug₩n--create₩n--file [file name]₩n--help₩n");
}

int verbose_flag;
int main (int argc, char **argv)
{
  int c;

  verbose_flag = 0;
  while (1)
    {
      static struct option long_options[] =
        {
          {"debug", no_argument,         0, 'd'},
          {"verbose", no_argument,       &verbose_flag, 1},
          {"create", no_argument,        0, 'c'},
          {"help",    no_argument,       0, 'h'},
          {"file",  required_argument,  0, 'f'},
          {0, 0, 0, 0}
        };
      /* getopt_long stores the option index here. */
      int option_index = 0;

      c = getopt_long (argc, argv, "dchf:",
                       long_options, &option_index);
```

```
/* Detect the end of the options. */
if (c == -1)
  break;

switch (c)
{
    case 0:
      break;

    case 'd':
      printf("debug mode₩n");
      break;

    case 'c':
      printf("create mode₩n");
      break;

    case 'f':
      printf ("file name is `%s'₩n", optarg);
      break;

    case 'h':
      help(argv[0]);
      break;
```

```
  case '?':
              help(argv[0]);
              break;
            default:
               help(argv[0]);
        }
    }
    if(verbose_flag)
      printf("verbos flag is set₩n");
    return 1;
}
```

Reference

- ✓ 리눅스 프로그램밍, 창병모, 생능출판
- ✓ https://www.44bits.io/ko/post/wsl2-install-and-basic-usage
- ✓ https://www.joinc.co.kr/w/man/3/getopt