6.10.1. Напишите программу *pwd*, определяющую полное имя текущего рабочего каталога. *#define U42* определяет файловую систему с длинными именами, отсутствие этого флага с короткими (14 символов).

```
/* Команда pwd.
 * Текст getwd() взят из исходных текстов библиотеки языка Си.
 */
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#define  ediag(e,r)      (e)
/*
 * getwd() возвращает полное имя текущего рабочего каталога.
 * При ошибке возвращается NULL, а в pathname копируется сообщение
 * об ошибке.
 */
#ifndef MAXPATHLEN
#define MAXPATHLEN      128
#endif

#define CURDIR          "."   /* имя текущего каталога      */
#define PARENTDIR       ".."  /* имя родительского каталога */
#define PATHSEP         "/"   /* разделитель компонент пути */
#define ROOTDIR         "/"   /* корневой каталог           */
#define GETWDERR(s)     strcpy(pathname, (s));
#define CP(to,from)     strncpy(to,from.d_name,DIRSIZ),to[DIRSIZ]='\0'

char *strcpy(char *, char *); char *strncpy(char *, char *, int);
char *getwd(char *pathname);
static char *prepend(char *dirname, char *pathname);

static int pathsize;                    /* длина имени */

#ifndef U42
char *getwd(char *pathname)
{
        char pathbuf[MAXPATHLEN];       /* temporary pathname buffer    */
        char *pnptr = &pathbuf[(sizeof pathbuf)-1]; /* pathname pointer */
        dev_t rdev;                     /* root device number           */
        int fil = (-1);                 /* directory file descriptor    */
        ino_t rino;                     /* root inode number            */
        struct direct dir;              /* directory entry struct       */
        struct stat d ,dd;              /* file status struct           */
                                        /* d - "."    dd - ".." | dname */
        char dname[DIRSIZ+1];           /* an directory entry           */

        pathsize = 0;
        *pnptr = '\0';
        if (stat(ROOTDIR, &d) < 0) {
                GETWDERR(ediag("getwd: can't stat /",
                               "getwd: нельзя выполнить stat /"));
                return (NULL);
        }
        rdev = d.st_dev; /* код устройства, на котором размещен корень */
        rino = d.st_ino; /* номер I-узла, представляющего корневой каталог */

        for (;;) {
                if (stat(CURDIR, &d) < 0) {
        CantStat:
                        GETWDERR(ediag("getwd: can't stat .",
                                       "getwd: нельзя выполнить stat ."));
                        goto fail;
                }
                if (d.st_ino == rino && d.st_dev == rdev)
                        break;          /* достигли корневого каталога */
                if ((fil = open(PARENTDIR, O_RDONLY)) < 0) {
                        GETWDERR(ediag("getwd: can't open ..",
                                       "getwd: нельзя открыть .."));
                        goto fail;
```

```
                }
                if (chdir(PARENTDIR) < 0) {
                        GETWDERR(ediag("getwd: can't chdir to ..",
                                       "getwd: нельзя перейти в .."));
                        goto fail;
                }
                if (fstat(fil, &dd) < 0)
                        goto CantStat;
                if (d.st_dev == dd.st_dev) {  /* то же устройство */
                        if (d.st_ino == dd.st_ino) {
                                /* достигли корня ".." == "." */
                                close(fil); break;
                        }
                        do {
                                if (read(fil, (char *) &dir,
                                    sizeof(dir)) < sizeof(dir)
                                ){
                        ReadErr:
                                  close(fil);
                                  GETWDERR(ediag("getwd: read error in ..",
                                                 "getwd: ошибка чтения .."));
                                        goto fail;
                                }
                        } while (dir.d_ino != d.st_ino);
                        CP(dname,dir);

                } else  /* ".." находится на другом диске: mount point */

                        do {
                                if (read(fil, (char *) &dir,
                                    sizeof(dir)) < sizeof(dir))
                                        goto ReadErr;
                                if( dir.d_ino == 0 )    /* файл стерт */
                                        continue;
                                CP(dname,dir);
                                if (stat(dname, &dd) < 0) {
                                        sprintf (pathname, "getwd: %s %s",
                                                 ediag ("can't stat",
                                            "нельзя выполнить stat"), dname);
                                        goto fail;
                                }
                        } while(dd.st_ino != d.st_ino ||
                                dd.st_dev != d.st_dev);
                close(fil);
                pnptr = prepend(PATHSEP, prepend(dname, pnptr));
        }

        if (*pnptr == '\0')             /* текущий каталог == корневому */
                strcpy(pathname, ROOTDIR);
        else {
                strcpy(pathname, pnptr);
                if (chdir(pnptr) < 0) {
                        GETWDERR(ediag("getwd: can't change back to .",
                                       "getwd: нельзя вернуться в ."));
                        return (NULL);
                }
        }
        return (pathname);

fail:
        close(fil);
        chdir(prepend(CURDIR, pnptr));
        return (NULL);
}

#else /* U42 */
extern char   *strcpy ();
extern DIR    *opendir();

char   *getwd (char *pathname)
{
    char    pathbuf[MAXPATHLEN];/* temporary pathname buffer */
    char   *pnptr = &pathbuf[(sizeof pathbuf) - 1];/* pathname pointer */
    char   *prepend ();         /* prepend dirname to pathname */
    dev_t rdev;                 /* root device number */
    DIR * dirp;                 /* directory stream */
    ino_t rino;                 /* root inode number */
    struct dirent  *dir;        /* directory entry struct */
    struct stat d,
                dd;             /* file status struct */

    pathsize = 0;
    *pnptr = '\0';
    stat (ROOTDIR, &d);
```

```
        rdev = d.st_dev;
        rino = d.st_ino;

        for (;;) {
            stat (CURDIR, &d);

            if (d.st_ino == rino && d.st_dev == rdev)
                break;              /* reached root directory */

            if ((dirp = opendir (PARENTDIR)) == NULL) {
                GETWDERR ("getwd: can't open ..");
                goto fail;
            }
            if (chdir (PARENTDIR) < 0) {
                closedir (dirp);
                GETWDERR ("getwd: can't chdir to ..");
                goto fail;
            }
            fstat (dirp -> dd_fd, &dd);
            if (d.st_dev == dd.st_dev) {
                if (d.st_ino == dd.st_ino) {
                /* reached root directory */
                    closedir (dirp);
                    break;
                }
                do {
                    if ((dir = readdir (dirp)) == NULL) {
                        closedir (dirp);
                        GETWDERR ("getwd: read error in ..");
                        goto fail;
                    }
                } while (dir -> d_ino != d.st_ino);
            }

            else
                do {
                    if ((dir = readdir (dirp)) == NULL) {
                        closedir (dirp);
                        GETWDERR ("getwd: read error in ..");
                        goto fail;
                    }
                    stat (dir -> d_name, &dd);
                } while (dd.st_ino != d.st_ino || dd.st_dev != d.st_dev);
            closedir (dirp);
            pnptr = prepend (PATHSEP, prepend (dir -> d_name, pnptr));
        }

        if (*pnptr == '\0')         /* current dir == root dir */
            strcpy (pathname, ROOTDIR);
        else {
            strcpy (pathname, pnptr);
            if (chdir (pnptr) < 0) {
                GETWDERR ("getwd: can't change back to .");
                return (NULL);
            }
        }
        return (pathname);

    fail:
        chdir (prepend (CURDIR, pnptr));
        return (NULL);
    }
    #endif

    /*
     * prepend() tacks a directory name onto the front of a pathname.
     */
    static char *prepend (
            register char *dirname,         /* что добавлять    */
            register char *pathname         /* к чему добавлять */
    ) {
            register int i;         /* длина имени каталога */

            for (i = 0; *dirname != '\0'; i++, dirname++)
                    continue;
            if ((pathsize += i) < MAXPATHLEN)
                    while (i-- > 0)
                            *--pathname = *--dirname;
            return (pathname);
    }

    #ifndef CWDONLY
    void main(){
            char buffer[MAXPATHLEN+1];
```

```
        char *cwd = getwd(buffer);
        printf( "%s%s\n", cwd ? "": "ERROR:", buffer);
}
#endif
```

6.10.2. Напишите функцию *canon*(), канонизирующую имя файла, т.е. превращающую его в **полное** имя (от корневого каталога), не содержащее компонент "." и "..", а также лишних символов слэш '/'. Пусть, к примеру, текущий рабочий каталог есть **/usr/abs/Cbook.** Тогда функция преобразует

```
.                   -> /usr/abs/C-book
..                  -> /usr/abs
../..               -> /usr
////..              -> /
/aa                 -> /aa
/aa/../bb           -> /bb
cc//dd/../ee        -> /usr/abs/C-book/cc/ee
../a/b/./d          -> /usr/abs/a/b/d
```

Ответ:

```
#include <stdio.h>
/* слэш, разделитель компонент пути */
#define SLASH       '/'
extern char *strchr (char *, char),
            *strrchr(char *, char);
struct savech{ char *s, c; };
#define SAVE(sv, str) (sv).s = (str); (sv).c = *(str)
#define RESTORE(sv) if((sv).s)   *(sv).s = (sv).c
/* Это структура для использования в таком контексте:
void main(){
  char *d = "hello"; struct savech ss;
  SAVE(ss, d+3); *(d+3) = '\0'; printf("%s\n", d);
  RESTORE(ss);                  printf("%s\n", d);
}
*/

/* ОТСЕЧЬ ПОСЛЕДНЮЮ КОМПОНЕНТУ ПУТИ */
struct savech parentdir(char *path){
   char *last  = strrchr( path, SLASH );
   char *first = strchr ( path, SLASH );
   struct savech sp; sp.s = NULL; sp.c = '\0';

if( last    == NULL  ) return sp; /* не полное имя    */
if( last[1] == '\0'  ) return sp; /* корневой каталог */
if( last    == first ) /* единственный слэш: /DIR     */
    last++;
   sp.s = last; sp.c = *last; *last = '\0';
   return sp;
}
#define isfullpath(s)   (*s == SLASH)
/* КАНОНИЗИРОВАТЬ ИМЯ ФАЙЛА */
void canon(
     char *where, /* куда поместить ответ */
     char *cwd,   /* полное имя текущего каталога */
     char *path   /* исходное имя для канонизации */
){   char *s, *slash;
   /* Сформировать имя каталога - точки отсчета */
   if( isfullpath(path)){
       s = strchr(path, SLASH);   /* @ */
       strncpy(where, path, s - path + 1);
       where[s - path + 1] = '\0';
       /* или даже просто strcpy(where, "/"); */
       path = s+1; /* остаток пути без '/' в начале */
   } else strcpy(where, cwd);

   /* Покомпонентный просмотр пути */
   do{  if(slash = strchr(path, SLASH)) *slash = '\0';
   /* теперь path содержит очередную компоненту пути */
        if(*path == '\0' || !strcmp(path, ".")) ;
     /* то просто проигнорировать "." и лишние "///" */
        else  if( !strcmp(path, ".."))
              (void) parentdir(where);
        else{ int len = strlen(where);
           /* добавить в конец разделяющий слэш */
              if( where[len-1] != SLASH ){
                  where[len] = SLASH;
                  where[len+1] = '\0';
              }
              strcat( where+len, path );
              /* +len чисто для ускорения поиска
               * конца строки внутри strcat(); */
        }
        if(slash){ *slash = SLASH; /* восстановить */
```

```
                        path   = slash + 1;
                }
        } while (slash != NULL);
}
char cwd[256], input[256], output[256];
void main(){
/* Узнать полное имя текущего каталога.
 * getcwd() - стандартная функция, вызывающая
 * через popen() команду pwd (и потому медленная).
 */
  getcwd(cwd, sizeof cwd);
  while( gets(input)){
    canon(output, cwd, input);
    printf("%-20s -> %s\n", input, output);
  }
}
```

В этом примере (изначально писавшемся для MS DOS) есть "странное" место, помеченное /*@*/. Дело в том, что в DOS функция *isfullpath* была способна распознавать имена файлов вроде **C:\aaa\bbb**, которые не обязательно **начинаются** со слэша.

6.11. Мультиплексирование ввода-вывода.

Данная глава посвящена системному вызову *select*, который, однако, мы предоставляем вам исследовать самостоятельно. Его роль такова: он позволяет опрашивать **нес- колько** дескрипторов открытых файлов (или устройств) и как только в файле появляется новая информация – сообщать об этом нашей программе. Обычно это бывает связано с дескрипторами, ведущими к сетевым устройствам.

6.11.1.

```
/* Пример использования вызова select() для мультиплексирования
 * нескольких каналов ввода. Этот вызов можно также использовать
 * для получения таймаута.
 *     Вызов: войти на терминалах tty01 tty02 и набрать на каждом
 *            sleep 30000
 *     затем  на tty00 сказать           select /dev/tty01 /dev/tty02
 *     и вводить что-либо на терминалах  tty01 и tty02
 * Сборка:      cc select.c -o select -lsocket
 */
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>  /* fd_set, FD_SET, e.t.c. */
#include <sys/param.h>  /* NOFILE */
#include <sys/select.h>
#include <sys/time.h>
#include <sys/filio.h>  /* для FIONREAD */
#define max(a,b)        ((a) > (b) ? (a) : (b))

char buf[512];          /* буфер чтения                       */
int fdin, fdout;        /* дескрипторы каналов stdin, stdout */
int nready;             /* число готовых каналов              */
int nopen;              /* число открытых каналов             */
int maxfd = 0;          /* максимальный дескриптор            */
int nfds;               /* сколько первых дескрипторов проверять */
int f;                  /* текущий дескриптор                 */
fd_set set, rset;       /* маски                              */

/* таблица открытых нами файлов */
struct _fds {
        int fd;         /* дескриптор */
        char name[30];  /* имя файла  */
} fds[ NOFILE ] = { /* NOFILE - макс. число открытых файлов на процесс */
        { 0, "stdin" }, { 1, "stdout" }, { 2, "stderr" }
        /* все остальное - нули */
};
struct timeval timeout, rtimeout;

/* выдать имя файла по дескриптору */
char *N( int fd ){
        register i;
        for(i=0; i < NOFILE; i++)
                if(fds[i].fd == fd ) return fds[i].name;
        return "???";
}

void main( int ac, char **av ){
        nopen = 3;              /* stdin, stdout, stderr */
        for( f = 3; f < NOFILE; f++ ) fds[f].fd = (-1);
        fdin = fileno(stdin);   fdout = fileno(stdout);
        setbuf(stdout, NULL);   /* отмена буферизации */
```

```
        FD_ZERO(&set);          /* очистка маски */

        for(f=1; f < ac; f++ )
                if((fds[nopen].fd = open(av[f], O_RDONLY)) < 0 ){
                   fprintf(stderr, "Can't read %s\n", av[f] );
                   continue;
                } else {
                   FD_SET(fds[nopen].fd, &set );   /* учесть в маске */
                      maxfd = max(maxfd, fds[nopen].fd );
                   strncpy(fds[nopen].name, av[f], sizeof(fds[0].name) - 1);
                   nopen++;
                }

        if( nopen == 3 ){
                fprintf(stderr, "Nothing is opened\n");
                exit(1);
        }

        FD_SET(fdin, &set); /* учесть stdin */
           maxfd = max(maxfd, fdin );
        nopen -= 2;         /* stdout и stderr не участвуют в select */
        timeout.tv_sec = 10;    /* секунд */
        timeout.tv_usec = 0;    /* миллисекунд */

    /* nfds - это КОЛИЧЕСТВО первых дескрипторов, которые надо
     * просматривать. Здесь можно использовать
     *          nfds = NOFILE;  (кол-во ВСЕХ дескрипторов   )
     * или      nfds = maxfd+1; (кол-во = номер последнего+1)
     * ( +1 т.к. нумерация fd идет с номера 0, а количество - с 1).
     */
        nfds = maxfd + 1;
        while( nopen ){

           rset = set; rtimeout = timeout; /* копируем, т.к. изменятся */
    /* опрашивать можно FIFO-файлы, терминалы, pty, socket-ы, stream-ы */

           nready = select( nfds, &rset, NULL, NULL, &rtimeout );

    /* Если вместо &rtimeout написать NULL, то ожидание будет
     * бесконечным (пока не собьют сигналом)
     */
           if( nready <= 0 ){  /* ничего не поступило */
                fprintf(stderr, "Timed out, nopen=%d\n", nopen);
                continue;
           }
           /* опрос готовых дескрипторов */
           for(f=0; f < nfds; f++ )
                if( FD_ISSET(f, &rset)){  /* дескриптор f готов */
                        int n;

                        /* Вызов FIONREAD позволяет запросить
                         * число байт готовых к передаче
                         * через дескриптор.
                         */
                        if(ioctl(f, FIONREAD, &n) < 0)
                                perror("FIONREAD");
                        else printf("%s have %d bytes.\n", N(f), n);

                        if((n = read(f, buf, sizeof buf)) <= 0 ){
                eof:
                            FD_CLR(f, &set); /* исключить */
                            close(f); nopen--;
                            fprintf(stderr, "EOF in %s\n", N(f));

                        } else {

                            fprintf(stderr, "\n%d bytes from %s:\n", n, N(f));
                            write(fdout, buf, n);
                            if( n == 4 && !strncmp(buf, "end\n", 4))
                            /* ncmp, т.к. buf может не оканчиваться \0 */
                                goto eof;
                        }
                }
        }
        exit(0);
}
```