**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

по дисциплине «Операционные системы»

на тему: «Управление файловой системой»

Выполнила: студентка гр. ИТП-11

Лисицин Н. С.

Принял: преподаватель-стажёр Карась О.В.

Гомель 2022

**Цель:** изучить принципы работы файлового менеджера и утилиты для работы с ним.

**Задача:**

Разработать утилиту в соответствии со своим вариантом.

**Вариант 4:** управление доступом.

**Ход работы.**

Управление памятью — это процесс, с помощью которого компьютерная система распределяет ограниченный объем физической памяти между различными процессами, которым эта память требуется, одновременно оптимизируя производительность.

Для реализации этой утилиты был написана программа на языке программирования *C*. При первом запуске программы пользователя просят ввести количество блоков диска, размер блока диска, количество узлов диска. Далее открывается меню с основными функциями программы такие как. Пример главного меню программы указан на рисунке 1.

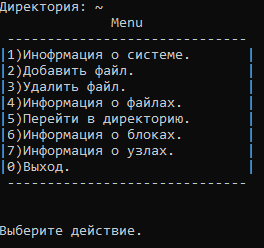


Рисунок 1 – Главное пеню программы

После выбора первого пункта меню на экран будет выведена информация о системе. Пример вывода информации представлен на рисунке 2.

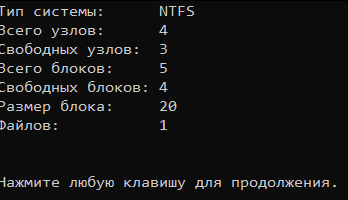


Рисунок 2 – Информация о системе

После этого пользователь может создать файл. В новом меню предлагается создание каталога или файла. Если при создании файла или каталога объем памяти будет меньше, чем количество свободного места на диске то файл не будет добавлен. Пример добавления нового файла указан на рисунке 3.

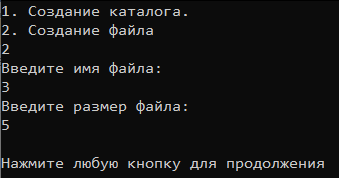


Рисунок 3 – Создание файла

После добавления нескольких файлов пользователь может открыть информацию о файлах, где указаны названия файлов. Пример вывода информации о файлах указан на рисунке 4.

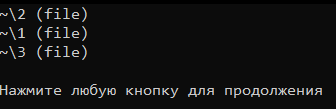


Рисунок 4 – Информация о созданных файлах

Так же можно посмотреть информацию о блоках и узлах. Вывод информации о блоках на экран приведен на рисунке 5.

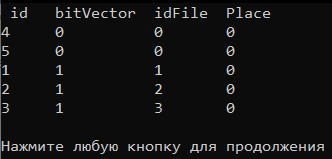


Рисунок 5 – Информация о блоках

**Вывод**: в ходе лабораторной работы были изучены основные утилиты для работы с файловым менеджером и разработана программа для управления доступом на языке программирования *C++*.

**ПРИЛОЖЕНИЕ А**

(обязательное)

Листинг программы

***Lab*11.*cpp*:**

#*include* <*stdio*.*h*>

#*include* <*conio*.*h*>

#*include* <*windows*.*h*>

#*include* <*stdlib*.*h*>

#*include* <*string*.*h*>

#*define* *SIZE* 10*u*

*char* *bufRus*[256];

*char*\* *Rus*(*const* *char*\**text*)

{

*CharToOem*(*text*,*bufRus*);

*return* *bufRus*;

}

*char* *path*[1000] = "~";

*struct* *Files*{

*int* *id*;

*int* *size*;

*int* *type*;

*char* \**name*;

*char* \**path*;

};

*int* *fileId* = 0;

*struct* *Blocks*{

*int* *id*;

*int* *bitVector*;

*int* *idFile*;

*int* *place*;

};

*int* *blockId* = 0;

*struct* *Nodes*{

*int* *id*;

*int* *bitVector*;

*int* *idFile*;

};

*int* *nodeId* = 0;

*struct* *SuperBlock*{

*char* *typeSystem*[5];

*int* *countBlocks*;

*int* *sizeBlock*;

*int* *countNodes*;

*Blocks* \**blocks*;

*int* *countFreeBlocks*;

*Nodes* \**nodes*;

*int* *countFreeNodes*;

*Files* \**files*;

*int* *countFiles*;

};

*int* *getpos*(*char* \**str*);

*void* *showInfo*(*SuperBlock* *superBlock*);

*void* *numerationNodes*(*SuperBlock* \**superBlock*);

*void* *numerationBlocks*(*SuperBlock* \**superBlock*);

*void* *printBlocks*(*SuperBlock* *superBlock*);

*void* *sortBlocks*(*SuperBlock* \**superBlock*);

*void* *printNodes*(*SuperBlock* *superBlock*);

*void* *sortNodes*(*SuperBlock* \**superBlock*);

*void* *addFile*(*SuperBlock* \**superBlock*);

*void* *showBlocks*(*SuperBlock* *superBlock*);

*void* *showNodes*(*SuperBlock* *superBlock*);

*void* *changeDirectory*(*SuperBlock* *superBlock*);

*void* *showFile*(*SuperBlock* *superBlock*);

*void* *deleteFile*(*SuperBlock* \**superBlock*);

*int* *main*()

{

*SuperBlock* *superBlock*;

*strcpy*(*superBlock*.*typeSystem*,"*NTFS*");

*do*{

*printf*(*Rus*("Введите количество блоков диска:\*n*"));

*scanf*("%*d*",&*superBlock*.*countBlocks*);

*printf*(*Rus*("\*n*Введите размер блока диска:\*n*"));

*scanf*("%*d*",&*superBlock*.*sizeBlock*);

*printf*(*Rus*("\*n*Введите количество узлов диска:\*n*"));

*scanf*("%*d*",&*superBlock*.*countNodes*);

*system*("*cls*");

} *while*(*superBlock*.*countBlocks* <= 0 || *superBlock*.*sizeBlock* <= 0 || *superBlock*.*countNodes* <= 0);

*superBlock*.*blocks* = (*struct* *Blocks*\*)*malloc*(*superBlock*.*countBlocks* \* *sizeof*(*Blocks*));

*superBlock*.*countFreeBlocks* = *superBlock*.*countBlocks*;

*numerationBlocks*(&*superBlock*);

*superBlock*.*nodes* = (*struct* *Nodes*\*)*malloc*(*superBlock*.*countNodes* \* *sizeof*(*Nodes*));

*superBlock*.*countFreeNodes* = *superBlock*.*countNodes*;

*numerationNodes*(&*superBlock*);

*superBlock*.*countFiles* = 0;

*int* *action*;

*while*(*true*){

*printf*(*Rus*("Директория: %*s*\*n*"),*path*);

*printf*(*Rus*(" *Menu*\*n*"));

*printf*(*Rus*(" ------------------------------\*n*"));

*printf*(*Rus*("|1)Инофрмация о системе. |\*n*"));

*printf*(*Rus*("|2)Добавить файл. |\*n*"));

*printf*(*Rus*("|3)Удалить файл. |\*n*"));

*printf*(*Rus*("|4)Информация о файлах. |\*n*"));

*printf*(*Rus*("|5)Перейти в директорию. |\*n*"));

*printf*(*Rus*("|6)Информация о блоках. |\*n*"));

*printf*(*Rus*("|7)Информация о узлах. |\*n*"));

*printf*(*Rus*("|0)Выход. |\*n*"));

*printf*(*Rus*(" ------------------------------\*n*"));

*printf*(*Rus*("\*n*\*n*Выберите действие.\*n*"));

*scanf*("%*d*",&*action*);

*system*("*cls*");

*switch*(*action*){

*case* 1:

*showInfo*(*superBlock*);

*break*;

*case* 2:

*addFile*(&*superBlock*);

*break*;

*case* 3:

*deleteFile*(&*superBlock*);

*break*;

*case* 4:

*showFile*(*superBlock*);

*break*;

*case* 5:

*changeDirectory*(*superBlock*);

*break*;

*case* 6:

*showBlocks*(*superBlock*);

*break*;

*case* 7:

*showNodes*(*superBlock*);

*break*;

*case* 0:

*return* 0;

*default*:

*printf*(*Rus*("Такого действия нет.\*n*"));

*printf*(*Rus*("Нажмите любую клавишу для продолжения.\*n*"));

*getch*();

}

*fflush*(*stdin*);

*system*("*cls*");

}

*getch*();

*return* 0;

}

*void* *addFile*(*SuperBlock* \**superBlock*){

*char* *name*[101], *choose*[2];

*int* *size*, *new*\_*node* = 1, *count*\_*block*;

*if*(*strcmp*(*path*,"~") == 0) *new*\_*node* = 0;

*int* *flag* = -1;

*printf*(*Rus*("1. Создание каталога.\*n*2. Создание файла\*n*"));

*scanf*("%1*s*",&*choose*);

*if*(*strcmp*(*choose*,"1") == 0){

*printf*(*Rus*("Введите имя каталога:\*n*"));

*scanf*("%100*s*",&*name*);

*if*(*getpos*(*name*) == 0 && *strlen*(*name*) != 0){

*flag* = 0;

}

*else* *printf*(*Rus*("Имя введено неверно\*n*"));

}

*if*(*strcmp*(*choose*,"2") == 0){

*printf*(*Rus*("Введите имя файла:\*n*"));

*scanf*("%100*s*",&*name*);

*if*(*getpos*(*name*) == 0 && *strlen*(*name*) != 0){

*printf*(*Rus*("Введите размер файла:\*n*"));

*scanf*("%*d*",&*size*);

*if*(*size* >= 0){

*if*(*size* / *superBlock*->*sizeBlock* \* *superBlock*->*sizeBlock* == *size*){

*count*\_*block* = *size* / *superBlock*->*sizeBlock*;

*if*(!*count*\_*block*) *count*\_*block*++;

}

*else* *count*\_*block* = *size* / *superBlock*->*sizeBlock* + 1;

*if*(*count*\_*block* <= *superBlock*->*countFreeBlocks*){

*flag* = 1;

} *else* *printf*(*Rus*("Недостаточно памяти\*n*"));

} *else* *printf*(*Rus*("Размер введён неверно\*n*"));

} *else* *printf*(*Rus*("Имя введено неверно\*n*"));

}

*if*(*flag* >= 0){

*if*(*superBlock*->*countFreeNodes* > 0 || *strcmp*(*path*,"~") != 0){

*if*(*superBlock*->*countFiles* == 0)

*superBlock*->*files* = (*struct* *Files*\*)*malloc*(1 \* *sizeof*(*Files*));

*else*

*superBlock*->*files* = (*struct* *Files*\*)*realloc*(*superBlock*->*files*,(*superBlock*->*countFiles* + 1) \* *sizeof*(*Files*));

*superBlock*->*files*[*superBlock*->*countFiles*].*id* = ++*fileId*;

*if*(*flag*){

*superBlock*->*files*[*superBlock*->*countFiles*].*type* = 1;

*superBlock*->*files*[*superBlock*->*countFiles*].*size* = *size*;

} *else* {

*superBlock*->*files*[*superBlock*->*countFiles*].*type* = 0;

*superBlock*->*files*[*superBlock*->*countFiles*].*size* = 0;

}

*superBlock*->*files*[*superBlock*->*countFiles*].*name* = *new* *char*[*strlen*(*name*)+1];

*strcpy*(*superBlock*->*files*[*superBlock*->*countFiles*].*name*,*name*);

*superBlock*->*files*[*superBlock*->*countFiles*].*path* = *new* *char*[*strlen*(*path*)+1];

*strcpy*(*superBlock*->*files*[*superBlock*->*countFiles*].*path*,*path*);

*strcat*(*superBlock*->*files*[*superBlock*->*countFiles*].*path*,"\\");

*superBlock*->*countFiles*++;

*if*(*flag*){

*for*(*int* *i* = 0; *i* < *count*\_*block*; *i*++){

*superBlock*->*blocks*[*i*].*bitVector* = 1;

*superBlock*->*blocks*[*i*].*idFile* = *fileId*;

*superBlock*->*blocks*[*i*].*place* = *i*;

*superBlock*->*countFreeBlocks*--;

}

*sortBlocks*(*superBlock*);

}

*if*(!*new*\_*node*){

*superBlock*->*nodes*[0].*bitVector* = 1;

*superBlock*->*nodes*[0].*idFile* = *fileId*;

*superBlock*->*countFreeNodes*--;

*sortNodes*(*superBlock*);

}

} *else* *printf*(*Rus*("Нет свободных узлов\*n*"));

}

*printf*(*Rus*("\*n*Нажмите любую кнопку для продолжения\*n*"));

*getch*();

}

*void* *showInfo*(*SuperBlock* *superBlock*){

*printf*(*Rus*("Тип системы: %*s*\*n*"),*superBlock*.*typeSystem*);

*printf*(*Rus*("Всего узлов: %*d*\*n*"),*superBlock*.*countNodes*);

*printf*(*Rus*("Свободных узлов: %*d*\*n*"),*superBlock*.*countFreeNodes*);

*printf*(*Rus*("Всего блоков: %*d*\*n*"),*superBlock*.*countBlocks*);

*printf*(*Rus*("Свободных блоков: %*d*\*n*"),*superBlock*.*countFreeBlocks*);

*printf*(*Rus*("Размер блока: %*d*\*n*"),*superBlock*.*sizeBlock*);

*printf*(*Rus*("Файлов: %*d*\*n*"),*superBlock*.*countFiles*);

*printf*(*Rus*("\*n*\*n*Нажмите любую клавишу для продолжения.\*n*"));

*getch*();

*system*("*cls*");

}

*void* *numerationNodes*(*SuperBlock* \**superBlock*){

*for*(*int* *i* = 0; *i* < *superBlock*->*countNodes*; *i*++){

*nodeId*++;

*superBlock*->*nodes*[*i*].*id* = *nodeId*;

*superBlock*->*nodes*[*i*].*bitVector* = 0;

*superBlock*->*nodes*[*i*].*idFile* = *NULL*;

}

}

*void* *numerationBlocks*(*SuperBlock* \**superBlock*){

*for*(*int* *i* = 0; *i* < *superBlock*->*countBlocks*; *i*++){

*blockId*++;

*superBlock*->*blocks*[*i*].*id* = *blockId*;

*superBlock*->*blocks*[*i*].*bitVector* = 0;

*superBlock*->*blocks*[*i*].*idFile* = *NULL*;

*superBlock*->*blocks*[*i*].*place* = *NULL*;

}

}

*void* *sortBlocks*(*SuperBlock* \**superBlock*){

*Blocks* *block*;

*for*(*int* *i* = 0; *i* < *superBlock*->*countBlocks*-1; *i*++){

*for*(*int* *j* = *i*+1; *j* < *superBlock*->*countBlocks*; *j*++){

*if*(*superBlock*->*blocks*[*j*].*bitVector* < *superBlock*->*blocks*[*i*].*bitVector*){

*block* = *superBlock*->*blocks*[*i*];

*superBlock*->*blocks*[*i*] = *superBlock*->*blocks*[*j*];

*superBlock*->*blocks*[*j*] = *block*;

}

}

}

*for*(*int* *i* = *superBlock*->*countBlocks*-1; *i* >= 1 && *superBlock*->*blocks*[*i*].*bitVector* != 0; *i*--){

*for*(*int* *j* = *i* - 1; *j* >= 0 && *superBlock*->*blocks*[*j*].*bitVector* != 0; *j*--){

*if*(*superBlock*->*blocks*[*j*].*idFile* > *superBlock*->*blocks*[*i*].*idFile*){

*block* = *superBlock*->*blocks*[*i*];

*superBlock*->*blocks*[*i*] = *superBlock*->*blocks*[*j*];

*superBlock*->*blocks*[*j*] = *block*;

}

}

}

*for*(*int* *i* = *superBlock*->*countBlocks*-1; *i* >= 1 && *superBlock*->*blocks*[*i*].*bitVector* != 0; *i*--){

*for*(*int* *j* = *i* - 1; *j* >= 0 && *superBlock*->*blocks*[*j*].*bitVector* != 0; *j*--){

*if*(*superBlock*->*blocks*[*i*].*idFile* == *superBlock*->*blocks*[*j*].*idFile* && *superBlock*->*blocks*[*j*].*place* > *superBlock*->*blocks*[*i*].*place*){

*block* = *superBlock*->*blocks*[*i*];

*superBlock*->*blocks*[*i*] = *superBlock*->*blocks*[*j*];

*superBlock*->*blocks*[*j*] = *block*;

}

}

}

}

*void* *sortNodes*(*SuperBlock* \**superBlock*){

*Nodes* *nodes*;

*for*(*int* *i* = 0; *i* < *superBlock*->*countNodes*-1; *i*++){

*for*(*int* *j* = *i*+1; *j* < *superBlock*->*countNodes*; *j*++){

*if*(*superBlock*->*nodes*[*j*].*bitVector* < *superBlock*->*nodes*[*i*].*bitVector*){

*nodes* = *superBlock*->*nodes*[*i*];

*superBlock*->*nodes*[*i*] = *superBlock*->*nodes*[*j*];

*superBlock*->*nodes*[*j*] = *nodes*;

}

}

}

*for*(*int* *i* = *superBlock*->*countNodes*-1; *i* >= 1 && *superBlock*->*nodes*[*i*].*bitVector* != 0; *i*--){

*for*(*int* *j* = *i* - 1; *j* >= 0 && *superBlock*->*nodes*[*j*].*bitVector* != 0; *j*--){

*if*(*superBlock*->*nodes*[*i*].*id* < *superBlock*->*nodes*[*j*].*id*){

*nodes* = *superBlock*->*nodes*[*i*];

*superBlock*->*nodes*[*i*] = *superBlock*->*nodes*[*j*];

*superBlock*->*nodes*[*j*] = *nodes*;

}

}

}

*for*(*int* *i* = 0; *i* < *superBlock*->*countNodes*-1 && *superBlock*->*nodes*[*i*].*bitVector* == 0; *i*++){

*for*(*int* *j* = *i* + 1; *j* < *superBlock*->*countNodes* && *superBlock*->*nodes*[*j*].*bitVector* == 0; *j*++){

*if*(*superBlock*->*nodes*[*i*].*id* > *superBlock*->*nodes*[*j*].*id*){

*nodes* = *superBlock*->*nodes*[*i*];

*superBlock*->*nodes*[*i*] = *superBlock*->*nodes*[*j*];

*superBlock*->*nodes*[*j*] = *nodes*;

}

}

}

}

*void* *showBlocks*(*SuperBlock* *superBlock*){

*printf*(" *id* *bitVector* *idFile* *Place*\*n*");

*for*(*int* *i* = 0; *i* < *superBlock*.*countBlocks*; *i*++){

*printf*("%-4*d* %-9*d* %-6*d* %-5*d*\*n*",*superBlock*.*blocks*[*i*].*id*,*superBlock*.*blocks*[*i*].*bitVector*,*superBlock*.*blocks*[*i*].*idFile*,*superBlock*.*blocks*[*i*].*place*);

}

*printf*(*Rus*("\*n*Нажмите любую кнопку для продолжения\*n*"));

*getch*();

}

*void* *showNodes*(*SuperBlock* *superBlock*){

*printf*(" *id* *bitVector* *idFile*\*n*");

*for*(*int* *i* = 0; *i* < *superBlock*.*countNodes*; *i*++){

*printf*("%-4*d* %-9*d* %-6*d*\*n*",*superBlock*.*nodes*[*i*].*id*,*superBlock*.*nodes*[*i*].*bitVector*,*superBlock*.*nodes*[*i*].*idFile*);

}

*printf*(*Rus*("\*n*Нажмите любую кнопку для продолжения\*n*"));

*getch*();

}

*void* *showFile*(*SuperBlock* *superBlock*){

*for*(*int* *i* = 0; *i* < *superBlock*.*countFiles*; *i*++){

*printf*("%*s*%*s*",*superBlock*.*files*[*i*].*path*, *superBlock*.*files*[*i*].*name*);

*if*(*superBlock*.*files*[*i*].*type*) *printf*(" (*file*)\*n*");

*else* *printf*(" (*katalog*)\*n*");

}

*printf*(*Rus*("\*n*Нажмите любую кнопку для продолжения\*n*"));

*getch*();

}

*void* *changeDirectory*(*SuperBlock* *superBlock*){

*char* \_*path*[1000], \_*path*2[1000];

*printf*(*Rus*("Перейти в директорию:\*n*"));

*scanf*("%*s*",&\_*path*);

*if*(*strcmp*("~",\_*path*) != 0){

*for*(*int* *i* = 0; *i* < *superBlock*.*countFiles*; *i*++){

*strcpy*(\_*path*2,*superBlock*.*files*[*i*].*path*);

*strcat*(\_*path*2,*superBlock*.*files*[*i*].*name*);

*if*(*strcmp*(\_*path*2,\_*path*) == 0 && *superBlock*.*files*[*i*].*type* == 0){

*strcpy*(*path*,\_*path*);

}

}

} *else*{

*strcpy*(*path*,"~");

}

}

*void* *deleteFile*(*SuperBlock* \**superBlock*){

*char* \_*path*[1000], \_*path*2[1000];

*int* *node*\_*flag* = 0;

*int* *flag*\_*del* = 1;

*int* *count* = 0, *flag* = 0;

*if*(*strcmp*(*path*,"~") == 0) *node*\_*flag* = 1;

*printf*(*Rus*("Введите имя удаляемого файла:\*n*"));

*scanf*("%*s*",&\_*path*2);

*strcpy*(\_*path*,*path*);

*strcat*(\_*path*,"\\");

*strcat*(\_*path*,\_*path*2);

*for*(*int* *i* = 0; *i* < *superBlock*->*countFiles*; *i*++){

*if*(*flag*\_*del*){

*strcpy*(\_*path*2,*superBlock*->*files*[*i*].*path*);

*strcat*(\_*path*2,*superBlock*->*files*[*i*].*name*);

*if*(*strcmp*(\_*path*,\_*path*2) == 0){

*flag*\_*del* = 0;

*flag* = 1;

}

} *else* {

*strcpy*(\_*path*2,*superBlock*->*files*[*i*].*path*);

*if*(*strncmp*(\_*path*,\_*path*2,*strlen*(\_*path*)) == 0) *flag* = 1;

}

*if*(*flag*){

*if*(*superBlock*->*files*[*i*].*type* == 1){

*for*(*int* *j* = *superBlock*->*countBlocks* - 1; *j* >= *superBlock*->*countFreeBlocks*; *j*--){

*if*(*superBlock*->*blocks*[*j*].*idFile* == *superBlock*->*files*[*i*].*id*){

*superBlock*->*blocks*[*j*].*bitVector* = 0;

*superBlock*->*blocks*[*j*].*idFile* = 0;

*superBlock*->*blocks*[*j*].*place* = 0;

*count*++;

}

}

*superBlock*->*countFreeBlocks* += *count*;

*count* = 0;

*sortBlocks*(*superBlock*);

}

*if*(*node*\_*flag* == 1){

*for*(*int* *j* = *superBlock*->*countNodes* - 1; *j* >= *superBlock*->*countFreeNodes*; *j*--){

*if*(*superBlock*->*nodes*[*j*].*idFile* == *superBlock*->*files*[*i*].*id*){

*superBlock*->*nodes*[*j*].*bitVector* = 0;

*superBlock*->*nodes*[*j*].*idFile* = 0;

*superBlock*->*countFreeNodes*++;

*node*\_*flag* == 0;

*break*;

}

}

*sortNodes*(*superBlock*);

}

*for*(*int* *j* = *i*+1; *j* < *superBlock*->*countFiles*; *j*++){

*superBlock*->*files*[*j*-1] = *superBlock*->*files*[*j*];

}

*superBlock*->*files* = (*struct* *Files*\*)*realloc*(*superBlock*->*files*,(--*superBlock*->*countFiles*) \* *sizeof*(*Files*));

*i*--;

*flag* = 0;

}

}

}

*void* *fileSort*2(*SuperBlock* \**superBlock*, *int* \**count*, *char* \**path*\_*file*){

*Files* *files*;

*char* *path*[1000];

*for*(*int* *i* = \**count*; *i* < *superBlock*->*countFiles*; *i*++){

}

}

*int* *getpos*(*char* \**str*){

*int* *pos* = 0;

*if*(*strchr*(*str*,'\\')) *pos* = *strchr*(*str*,'\\')-*str*+1;

*return* *pos*;

}