|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБАРАТОРНОЙ РАБОТЫ №2**

по дисциплине: «Операционные системы»

|  |  |  |
| --- | --- | --- |
| Студент |  | Беркович Евгений Сергеевич |
| Группа |  | РК6-55Б |
| Тип задания |  | Обработка каталогов файловых систем |
| Вариант |  | 02 |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Беркович Е.С.**\_ \_

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Грошев С. В.** \_\_

*подпись, дата фамилия, и.о.*

*Москва, 2023 г.*

**Техническое задание**

Требуется разработать программу для визуального просмотра оглавлений каталогов файловых систем OS UNIX, которая позволяет определить состав любого доступного каталога. Имя каталога должно передаваться программе через аргумент командный строки её вызова. В результате обработки оглавления заданного каталога программа должна обеспечивать стандартный вывод файлов заданного каталога, в порядке возрастания даты изменения соответствующего файла. При разработке программы необходимо использовать библиотечные функции просмотра каталогов.

**Приложение №1**

**Содержимое файла directory.c**

|  |
| --- |
| #include <dirent.h>  #include <unistd.h>  #include <errno.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/stat.h>  #include <unistd.h>  typedef struct FileInfo {  char\* name;  unsigned long modif\_time;  } FileInfo;  int scanfolder(DIR\*,FileInfo\*\*,size\_t\*);  void diag(const char\*);  int compare\_for\_qsort(const void \* a, const void \* b){  FileInfo\* data\_1 = (FileInfo \*)a;  FileInfo\* data\_2 = (FileInfo \*)b;  return ( data\_1->modif\_time - data\_2->modif\_time);  }  void diag(const char\* mes){  write(1, mes, strlen(mes));  write(1, ": ", 2);  write(1, strerror(errno), strlen(strerror(errno)));  write(1, "\n", 1);  }  int scanfolder(DIR\* folder\_ptr,FileInfo\*\* folder\_contents,size\_t\* contents\_count){  struct dirent\* folder;  struct stat sbuf;  (\*folder\_contents) = (FileInfo\*)malloc(sizeof(FileInfo) \* (\*contents\_count));  FileInfo\* temp\_ptr = NULL;  while((folder = readdir(folder\_ptr)) != NULL) {  stat(folder->d\_name, &sbuf);  switch (sbuf.st\_mode & \_\_S\_IFMT)  {  case \_\_S\_IFDIR:  (\*folder\_contents)[(\*contents\_count)-1].name = (char\*)malloc(sizeof(char) \* strlen(folder->d\_name)+2);  memcpy((\*folder\_contents)[(\*contents\_count)-1].name, folder->d\_name, strlen(folder->d\_name));  (\*folder\_contents)[(\*contents\_count)-1].name[strlen(folder->d\_name)]='/';  (\*folder\_contents)[(\*contents\_count)-1].name[strlen(folder->d\_name)+1]='\0';  break;  default:  (\*folder\_contents)[(\*contents\_count)-1].name = (char\*)malloc(sizeof(char) \* strlen(folder->d\_name)+1);  memcpy((\*folder\_contents)[(\*contents\_count)-1].name, folder->d\_name, strlen(folder->d\_name)+1);  break;  }    (\*folder\_contents)[(\*contents\_count)-1].modif\_time=sbuf.st\_mtim.tv\_sec;  (\*contents\_count) += 1;  if ((temp\_ptr = (FileInfo\*)realloc((\*folder\_contents),(\*contents\_count) \* sizeof(FileInfo))) != NULL){  (\*folder\_contents) = temp\_ptr;  } else {  return 127;  }  }  return(0);  }  char\* get\_folder\_name(int argc, char\* argv[]){  char\* fold\_name = NULL;  const char\* default\_fold\_name = ".";  size\_t fold\_name\_len;  switch (argc)  {  case 1:  fold\_name\_len=strlen(default\_fold\_name)+1;  fold\_name = (char\*)malloc(fold\_name\_len\*(sizeof(char)));  memcpy(fold\_name,default\_fold\_name,fold\_name\_len);  break;  default:  fold\_name\_len=strlen(argv[1])+1;  fold\_name = (char\*)malloc(fold\_name\_len\*(sizeof(char)));  memcpy(fold\_name,argv[1],fold\_name\_len);  break;  }  return fold\_name;  }  void output(FileInfo\* folder\_contents, size\_t contents\_count){  size\_t blanketing = 20;  for(size\_t i = 0; i < contents\_count; i++){  write(1,folder\_contents[i].name,strlen(folder\_contents[i].name));  for(size\_t whitespace\_count=0; whitespace\_count < blanketing-strlen(folder\_contents[i].name); whitespace\_count++){  write(1," ",1);  }  //write(1,&folder\_contents[i].modif\_time,sizeof(folder\_contents[i].modif\_time));  fprintf(stderr,"%ld",folder\_contents[i].modif\_time);  write(1,"\n",1);  }  }  int main(int argc, char\* argv[]) {  char\* folder\_name = get\_folder\_name(argc,argv);  DIR\* folder\_ptr = NULL;  FileInfo\* folder\_contents = NULL;  size\_t contents\_count=1;  if((folder\_ptr = opendir(folder\_name)) != NULL){  scanfolder(folder\_ptr,&folder\_contents,&contents\_count);  closedir(folder\_ptr);  qsort(folder\_contents, contents\_count - 1, sizeof(FileInfo), compare\_for\_qsort);  output(folder\_contents, contents\_count - 1);  for (size\_t i = 0; i < contents\_count - 1; i++)  {  free(folder\_contents[i].name); |

**Приложение №2**

**Результат работы программы:**

Программа принимает аргумент командной строки: каталог, который мы хотим просмотреть. Она отображает все файлы в отсортированном по дате изменения виде.

