Работа с файловой системой

Типичные задачи

- Просмотр содержимого каталогов
- Рекурсивный обход всех каталогов
- Получение информации о состоянии файла

Обобщенный интерфейс

- Каталог хранит только имена файлов
- Остальная информация о файле хранится в индексном дескрипторе

Каталоги. Открытие/закрытие

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir(const char *name);
int  closedir(DIR *dir);
```

DIR - структура, используемая для работы с каталогом (дескриптор каталога)

Каталоги. Чтение содержимого

```
#include <sys/types.h>
#include <dirent.h>

struct dirent *readdir(DIR *dir);
```

- Может возвращать `.` и `..`
- Порядок следования имен вложенных файлов не определен
- Возвращаемое значение размещается внутри дескриптора `dir`
- Поля `struct dirent`
  - `char d_name[]` - имя вложенного файла
  - `ino_t d_ino` - использовать не рекомендуется

Каталоги. Позиционирование

```
#include <sys/types.h>
#include <dirent.h>

off_t telldir(DIR *dir);
void  seekdir(DIR *dir, off_t offset);
```

Метаданные файла. Получение

```
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *file_name, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *file_name, struct stat *buf);
```

Обработка типа файла

```
struct stat
{
  dev_t   st_dev;   /* устройство */
  ino_t   st_ino;   /* индексный дескриптор (inode) */
  mode_t  st_mode;  /* права доступа, флаги и тип файла*/
  nlink_t st_nlink; /* число жестких связей */
  uid_t   st_uid;   /* идентификатор владельца */
  gid_t   st_gid;   /* идентификатор группы */
  ...
} s;
```

Макросы определения типа файла

```
S_ISDIR(s.st_mode), S_ISREG(s.st_mode),
S_ISLNK(s.st_mode), ...
```

"Ручная" обработка (для проверки на тип каталог)

```
s.st_mode & S_IFMT == S_IFDIR
```

Пример обработки каталога

Задача

Написать программу, выводящую на экран имена всех символических ссылок каталога

Календарное время

Включает год, месяц, день месяца, время суток

- Тип `time_t`
- POSIX: календарное время - количество секунд от "начала эпохи"
- UNIX:
  - начало эпохи - полночь 01.01.1970 по Гринвичу
  - отсчет ведется по Гринвичу без перехода на летнее время (Coordinated Universal Time - UTC)

Работа с календарным временем

ANSI C

```
#include <time.h>
// объявление time_t
struct tm
{
    int tm_sec; /* секунды [0-60]*/
    int tm_min; /* минуты [0-59]*/
    int tm_hour; /* час [0-23]*/
    int tm_mday; /* день месяца [1-31]*/
    int tm_mon; /* месяц [0-11], 0 = январь */
    int tm_year; /* число лет от 1900 г: 100 => 2000 г. */
    int tm_wday; /* день недели [0-6],0 = Вс */
    int tm_yday; /* день года [0-365] */
    int tm_isdst; /* действует ли летнее время:
                     >0 - да; 0 - нет; <0 - неизв. */
};
```

Функции (1)

Получение текущего времени

```
time_t time(time_t *pVal);
```

Функции (2)

Конвертация времени

- Время по UTC

```
struct tm *gmtime(const time_t *timep);
```

- Время в определенном часовом поясе

```
struct tm *localtime(const time_t *timep);
```

```
extern char *tzname[2];
   /* [0] - имя ч.п. зимнего времени
      [1] - имя ч.п. летнего времени */
extern long timezone;
   /* разница в секундах поясного времени и UTC */
```

Функции (3)

Конвертация времени

```
time_t mktime(struct tm *timeptr);
```

- Поля `tm_wday`, `tm_yday` при расчете игнорируются
- Для корректной обработки летнего времени `tm_isdst` требуется установить в `-1`
- Нормализация остальных полей

  40 октября -> 9 ноября
- Обновление значений полей `tm_wday`, `tm_yday`, `tm_isdst`
- Обновление `tzname`

Функции (4)

Строковое представление

```
char *asctime(const struct tm *timeptr);
char *ctime(const time_t *timep);
```

Получение строки вида "Tue Nov 7 11:32:11 2000\n"

```
size_t strftime(char *s, size_t max,
                const char *format,
                const struct tm *tm);
```