

# Системные вызовы и библиотеки Unix SVR4

## Введение (продолжение)

Иртегов Д.В.  
ФФ/ФИТ НГУ

Электронный лекционный курс подготовлен в рамках реализации  
Программы развития НИУ-НГУ на 2009-2018 г.г.

# Организация практикума

37 задач (<http://ccfit.nsu.ru/~fat/svr4tasks-new.html>)

- На 5 необходимо сдать 23 задачи, из них 10 из второй части курса
- На 4 — 11 задач
- На 3 — 6 задач
- Задачи надо сдавать на ОС Solaris ([solarka.ccfit.nsu.ru](http://solarka.ccfit.nsu.ru) или 10.4.0.68).

# Что такое Solaris

- ОС, основанная на ядре Unix System V Release 4.
- Прямой наследник оригинального AT&T Unix
- Разрабатывается компанией Oracle (ранее Sun Microsystems)
- Официально поддерживает процессоры x86/x64, SPARC v9
- Есть бесплатная версия с открытыми исходниками:  
<https://www.openindiana.org/>  
<https://github.com/illumos/illumos-gate>

# Чем Solaris отличается от Linux

- Solaris был раньше :)
- Linux — лицензионно чистый клон Unix, распространяемый под лицензией GPL
- Обе системы поддерживают стандартизованный API POSIX/X-Open, но Solaris сертифицирован, а Linux нет.
- Системы имеют разные ABI (бинарно несовместимы)

# Рекомендованная литература

- К. Хевиленд, Д. Грей, Б. Салама, Системное программирование в UNIX. Руководство программиста по разработке ПО (есть в библиотеке)
- А. Робачевский Операционная система UNIX
- У. Р. Стивенс, С. А. Раго UNIX Профессиональное программирование
- R. McDougal, J. Mauro Solaris Internals (для продвинутых)

# Рекомендованная литература

- Man(1)
  - Встроенное системное руководство, доступно из командной строки по команде `man`
- Oracle Solaris 10/11 documentation  
<http://www.oracle.com/technetwork/documentation/solaris-10-192992.html> ,  
бывш. docs.sun.com  
Там же есть и все man-страницы в html с гиперссылками и поиском
- Исходные тексты: <https://github.com/illumos/illumos-gate>

# man(1)

- Формат команды  
man strcpu  
или  
man -s 2 write
- Поиск по заголовкам  
apropos write
- Номера секций
  - 1 — команды shell
  - 1M — команды shell, доступные администратору
  - 2 — системные вызовы
  - 3C — стандартная библиотека C

# Формат страницы man

ЗАГОЛОВОК (TITLE) - обычно имя библиотечной функции или системного вызова

- . СЕКЦИЯ (SECTION) - раздел Руководства
- . БИБЛИОТЕКА (LIBRARY) - для секции 3, библиотечных функций, одно из C, S, M, E, X или G.
- . ИМЯ (NAME) - имя и краткое описание системного вызова или библиотечной функции (в одной строке)
- . ИСПОЛЬЗОВАНИЕ (SYNOPSIS) - как вызвать системный вызов/библиотечную функцию.
- . ОПИСАНИЕ (DESCRIPTION) - описывает работу системного вызова или функции.
- . ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ (RETURN VALUE) - как интерпретировать возвращаемый код.
- . СМ. ТАКЖЕ (SEE ALSO) - страницы Руководства, имеющие отношение к этой странице.
- . АТТРИБУТЫ (ATTRIBUTES) - каким стандартам соответствует функция, thread-safety, поддержка длинных файлов и т.д.

Кроме того, страница Руководства может содержать разделы:

- . ПРИМЕРЫ (EXAMPLES)
- . ФАЙЛЫ (FILES)
- . СООБЩЕНИЯ (DIAGNOSTICS)
- . ЗАМЕЧАНИЯ (NOTES)
- . (ПРЕДУПРЕЖДЕНИЯ) WARNINGS
- . (ОШИБКИ) BUGS
- . (ПРОБЛЕМЫ) CAVEATS



# Пример страницы руководства

## ИМЯ

`perror` - напечатать системное сообщение об ошибке

## ИСПОЛЬЗОВАНИЕ

```
#include <stdio.h>

void perror(const char *s);
```

## ОПИСАНИЕ

`perror` выводит в стандартный вывод диагностики сообщение, описывающее последнюю ошибку, обнаруженную при вызове системной или библиотечной функции. Сначала печатается строка параметра `s`, затем двоеточие и пробел, затем сообщение и перевод строки.

... Номер ошибки берется из внешней переменной `errno`, которая устанавливается при ошибочном, но не очищается при успешном системном вызове.

## СМ. ТАКЖЕ

`intro(2)`, `fmtmsg(3C)`, `strerror(3C)`.

# Пример страницы руководства

## ИМЯ

time - получить время

## ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <time.h>
time_t time(time_t *tloc);
```

## ОПИСАНИЕ

time возвращает значение времени в секундах от 1 января 1970, 00:00:00 UTC. Если tloc не равен нулю, возвращаемое значение будет также сохранено в переменной, на которую указывает tloc.

## ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

При успешном завершении, time возвращает значение времени.

Иначе возвращается значение -1 и значение errno отражает причину ошибки.

## СМ. ТАКЖЕ

ctime(3C), stime(2).

# Примеры использования time(2)

## ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ time(2)

- . объявления и инициализации

```
#include <sys/types.h>
#include <time.h>
time_t t1, t2, t3, t4;
time_t *tp = &t2;
```

- . нулевое значение параметра time(2) означает, что должно использоваться возвращаемое значение

```
t1 = time ( 0 );
```

- . инициализированный указатель в качестве аргумента

```
t3 = time ( tp );
```

- . адрес переменной в качестве аргумента

```
(void) time ( &t4 );
```

# Пример страницы руководства

## ИМЯ

ctime, localtime, gmtime, asctime, tzset - преобразование даты и времени в строку

## ИСПОЛЬЗОВАНИЕ

```
#include <time.h>
char * ctime(const time_t *clock);
struct tm * localtime(const time_t *clock);
struct tm * gmtime(const time_t *clock);
char * asctime(const struct tm * tm);
extern time_t timezone, altzone;
extern int daylight;
extern char *tzname[2];
void tzset(void);
```

## ОПИСАНИЕ

ctime, localtime и gmtime получают аргумент типа указатель на time\_t, представляющий время в секундах с 00:00:00 UTC 1 января 1970. ctime возвращает указатель на строку из 26 символов следующего вида ...

```
Sun Sep 16 01:03:52 1973\n\0
```

# Пример страницы руководства (продолжение)

... Объявление всех функций и внешних переменных, а также описание структуры `tm`, содержатся в файле `<time.h>`. Описание структуры выглядит так:

```
struct tm {
    int tm_sec;    /* seconds after minute [0,61] */
    int tm_min;    /* minutes after the hour [0,59] */
    int tm_hour;   /* hour since midnight [0,23] */
    int tm_mday;   /* day of the month [1,31] */
    int tm_mon;    /* months since January [0,11] */
    int tm_year;   /* years since 1900 */
    int tm_wday;   /* days since Sunday [0,6] */
    int tm_yday;   /* days since January 1 [0,365] */
    int tm_isdst;  /* flag for alternate daylight savings time */
};
```