Разработка программного обеспечения ОС UNIX

Учет стандартов и ограничений системы. Библиотека Си.

Системные вызовы.

Настройка программы на стандарт

Проверка версии реализованного в системе стандарта

<unistd.h>

#if …==…

_POSIX_SOURCE	//POSIX.1-1990
_POSIX_C_SOURCE	//POSIX.1-1990 и POSIX.1-1992
_POSIX_C_SOURCE==199309L	// POSIX.1b-1993
_POSIX_C_SOURCE==199506L	// POSIX.1b-1996
_POSIX_C_SOURCE==200112L	// POSIX.1-2001
_XOPEN_SOURCE == 700	//XPG4, SUSv4
_XOPEN_SOURCE_EXTENDED == 1	
_POSIX_VERSION == 200809L	//POSIX.1-2008
_POSIX2_VERSION == 200809L	
_XOPEN_VERSION == 700	//X/Open Portability Guide

Компиляция программы с соответствующими модели опциями

Из командной строки – утилита getconf:

```
c99 $(getconf POSIX_V6_LP64_OFF64_CFLAGS) \
-D_XOPEN_SOURCE=600 \
$(getconf POSIX_V6_LP64_OFF64_LDFLAGS) foo.c -o foo \
$(getconf POSIX_V6_LP64_OFF64_LIBS) –lxnet
```

Подготовка опций в программе – функция confstr():

```
size_t confstr(      int name,
                     char *buf,
                     size_t len);
n =  confstr(_CS_XBS5_ILP32_OFF32_LIBS, buf, 4096);
```

Модели программирования

_CS_POSIX_V7_ILP32_OFF32_CFLAGS
_CS_POSIX_V7_ILP32_OFFBIG_CFLAGS
_CS_POSIX_V7_LP64_OFF64_CFLAGS
_CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS
_CS_POSIX_V7_THREADS_CFLAGS

……..._LDFLAGS
……..._LIBS

I	int
L	long
P	* (pointer)
OFF	off_t

Определение ограничений системы в программе

<limits.h>

ARG_MAX	Maximum length of argument to the exec functions including environment data.
LOGIN_NAME_MAX	Maximum length of a login name.
OPEN_MAX	A value one greater than the maximum value that the system may assign to a newly-created file descriptor
PAGESIZE	Size in bytes of a page.
PTHREAD_THREADS_MAX	Maximum number of threads that can be created per process.
SEM_VALUE_MAX	The maximum value a semaphore may have.
TIMER_MAX	Maximum number of timers per process supported by the implementation.

и т.п.

Динамическое определение ограничений

include <unistd.h>

long sysconf(int name);

_SC_ARG_MAX

_SC_TIMER_MAX

_SC_RTSIG_MAX …

long fpathconf(int fildes, int name);

long pathconf(const char *path, int name);

_PC_PATH_MAX

_PC_LINK_MAX

_PC_SYMLINK_MAX

_PC_CHOWN_RESTRICTED

Системные вызовы. Обработка ошибок

Системный вызов оформлен как вызов функции.

Определение успешности:

1. Анализ возвращаемого значения.
2. В случае ошибки – анализ errno. Если вызов прерван сигналом – требуется его перезапуск. Далее см. п. 1.
3. При необходимости – печать сообщения.

Пример. Функция (системный вызов) read. Некоторые значения errno.

EINTR	прерван сигналом
EIO	ошибка ввода/вывода
EINVAL	ошибочный аргумент
ENOMEM	недостаточно памяти для операции
ENOBUFS	системе не хватает ресурсов
EISDIR	попытка чтения из файла, являющегося каталогом
EAGAIN	необходимо повторить операцию (неблокирующий режим)

Сообщение об ошибке

<errno.h> макроопределения и коды ошибок

```
#include <stdio.h>
void perror(const char *s);
```

Выдает на терминал:

Сообщение пользователя *:сообщение об ошибке по errno\n*

Пример:

```
if ((bufptr = malloc(szbuf)) == NULL) {
    perror("malloc"); exit(2);
}
```

Получить сообщение об ошибке в памяти:

```
#include <string.h>
char *     strerror     ( int errnum );
char *     strerror_l   ( int errnum, locale_t locale );
int        strerror_r   ( int errnum, char *strerrbuf, size_t buflen );
```

Аварийное или немедленное завершение программы

Аварийное завершение

```
#include <stdlib.h>
void abort(void);
```

Возможно создание файла core.

```
#include <stdlib.h>
void _Exit(int status);
```

```
#include <unistd.h>
void _exit(int status);
```

```
#include <stdlib.h>
void exit(int status);
```

Вызывает функции, зарегистрированные с помощью

```
int atexit(void (*func)(void));
```

Ловим ошибки…

```
#include <assert.h>
void assert(scalar expression);
```

Реализация:

```
#ifndef NDEBUG
#define assert(ex) { \
   if ( !ex ) { \
     fprintf( stderr, “Assertion failed: file \”%s\”, line %d\n, __FILE__, __LINE__);\
     abort(); \
   }}
#endif
```

Пример:

```
//#define NDEBUG
#include <assert.h>
…
     ptr = malloc(…);
     assert( ptr );
```

Некоторые полезные функции

Измерение времени (грубо, но переносимо):

```
#include <time.h>
clock_t clock(void);

    clock_t start = clock();
    …
    clock_t end = clock();
    double runtime = (double)( end – start ) / (double)
    CLOCKS_PER_SEC;
```

32-разрядная: переполнение за 2147 секунды или 36 минут.

Точное измерение (advanced realtime) – не везде реализовано.

```
#include <sys/types.h>
#include <time.h>
int clock_getcpuclockid(pid_t pid, clockid_t *clock_id);
```

Измерение времени в Solaris (cpc) – perf.counters

```
#include <inttypes.h>   <stdlib.h>          <stdio.h> <unistd.h>              <libcpc.h>
Int main(int argc, char *argv[]) {
int cpuver, iter;        char *setting = NULL;              cpc_event_t event;
    if (cpc_version(CPC_VER_CURRENT) != CPC_VER_CURRENT)  error("application:library cpc version
    mismatch!");
    if ((cpuver = cpc_getcpuver()) == -1)  error("no performance counter hardware!");
    if ((setting = getenv("PERFEVENTS")) == NULL) setting = "pic0=EC_ref,pic1=EC_hit";
    if (cpc_strtoevent(cpuver, setting, &event) != 0) error("can't measure '%s' on this processor", setting);
    setting = cpc_eventtostr(&event);
    if (cpc_access() == -1) error("can't access perf counters: %s", strerror(errno));
    if (cpc_bind_event(&event, 0) == -1) error("can't bind lwp%d: %s", _lwp_self(), strerror(errno));
    for (iter = 1; iter <= 20; iter++) {
            cpc_event_t before, after;
            if (cpc_take_sample(&before) == -1) break;
            /*  Что-то вычисляем  */
            if (cpc_take_sample(&after) == -1) break;
            (void) printf("%3d: %" PRId64 " %" PRId64 ", iter,
                        after.ce_pic[0] - before.ce_pic[0],
                        after.ce_pic[1] - before.ce_pic[1]);
    }
    if (iter != 20)
    error("can't sample '%s': %s", setting, strerror(errno));
    free(setting);
    return (0);
}
```

Разбор аргументов командной строки

```
#include <unistd.h>
int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int opterr, optind, optopt;
```

Пример:

```
int c;        char *filename;
extern char *optarg; extern int optind, optopt, opterr;
while ((c = getopt(argc, argv, ":abf:")) != -1) {
   switch(c) {
   case 'a':   printf("a is set\n");                                    break;
   case 'b':   printf("b is set\n");                                    break;
   case 'f':    filename = optarg;  printf("filename is %s\n", filename);   break;
   case ':':    printf("-%c without filename\n", optopt);              break;
   case '?':   printf("unknown arg %c\n", optopt);                     break;
   }
}
Linux:  getopt_long( argc, argv, optstr, longopt, NULL );   //«Длинные опции»: --help
     struct longopt { {“help”, 0, NULL, ‘h’}, … ,
                          {NULL, 0, NULL, 0}};
```

Работа с учетной информацией о пользователях

```
#include <pwd.h>
struct passwd *getpwnam(const char *);
struct passwd *getpwuid(uid_t);

void setpwent(void);
struct passwd *getpwent(void);
void endpwent(void);
struct passwd {
    char    *pw_name;       //User's login name.
    uid_t    pw_uid;        //Numerical user ID.
    gid_t    pw_gid;        //Numerical group ID.
    char    *pw_dir;        //Initial working directory.
    char    *pw_shell;      //Program to use as shell.
};
```

Работа с учетной информацией о группах

```
#include <grp.h>
struct group {
    char   *gr_name;        //The name of the group.
    gid_t   gr_gid;         //Numerical group ID.
    char  **gr_mem;         //Pointer to a null-terminated array of character
};                          //  pointers to member names.
struct group  *getgrgid(gid_t);
struct group  *getgrnam(const char *);
void          setgrent(void);
struct group  *getgrent(void);
void          endgrent(void);
```

Переменные окружения и др.

```
#include <stdlib.h>

char *getenv(const char *name);
int setenv(const char *envname, const char *envval, int overwrite);
int unsetenv(const char *name);

int system(const char *command);

int rand_r(unsigned *seed);
int rand(void);
void srand(unsigned seed);
double drand48(void);
unsigned short *seed48(unsigned short seed16v[3]);

#include <unistd.h>
char *getlogin(void);
```

Формула генератора

Xn+1 = (aXn + c)mod m n>= 0

Функции с переменным числом аргументов

```
#include <stdarg.h>
void va_start(va_list ap, argN);
void va_copy(va_list dest, va_list src);
type va_arg(va_list ap, type);
void va_end(va_list ap);
```

Возможная реализация:

```
#define va_start( ap, parm )        (ap)=(char*)(&(parm)+1)
#define va_arg( ap, type )          ((type*)((char*)(ap)+=sizeof(type)))[-1]
```

Применение:

```
int function_with-varargs( const char *args, ...) {
    va_list ap;
    va_start(ap, args);
    arg1 = va_arg(ap, int);
    arg2 = va_arg(ap, double);
    va_end(ap);
```