05. Работа над проектом. Дополнительные темы.

Егор Орлов

Курс: UNIX-DEV-SYS. Системное программирование в среде UNIX (Linux/FreeBSD). ВИШ СПбПУ, 2021

Содержание

1. Организация вывода HTML-документов

1.1. Использование системных вызовов ввода-вывода (read/write)

1.1.1. Код lesson5: examples/readwrite.c

```
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>

const char *myfile = "index.html";

int main() {
        int fdin;
        struct stat statbuf;
```

```
        char *bufin;
        if ((fdin=open(myfile, O_RDONLY)) < 0) {
                perror(myfile);
                return 1;
        }
        if (fstat(fdin, &statbuf) < 0) {
                perror(myfile);
                return 1;
        }
    if ((bufin = (char *)malloc(statbuf.st_size)) == NULL) {
                fprintf(stderr, "Error alocating memory!\n");
                return 1;
        }
        if (read(fdin,bufin,statbuf.st_size) != statbuf.st_size) {
                perror(myfile);
                return 1;
        }
        if (write(1,bufin,statbuf.st_size) != statbuf.st_size) {
                perror("stdout");
                return 1;
        }
        free(bufin);
        close(fdin);
        return 0;
}
```

1.1.2. Получение индексного дескриптора файла

- По файловому дескриптору

```
#include <sys/stat.h>
int fstat(int fildes, struct stat *buf);
```

- По имени файла

```
#include <sys/stat.h>
int stat(const char *path, struct stat *buf);
```

- Для символической ссылки

```
#include <sys/stat.h>
int lstat(const char *path, struct stat *buf);
```

  Для символической ссылки - возвращает дескриптор самой ссылки, а не куда она указывает.

- Структура индексного дескриптора

```
struct stat {
    dev_t         st_dev;      /* устройство */
    ino_t         st_ino;      /* inode */
    mode_t        st_mode;     /* режим доступа */
    nlink_t       st_nlink;    /* количество жестких ссылок */
```

```
    uid_t         st_uid;      /* идентификатор пользователя-владельца */
    gid_t         st_gid;      /* идентификатор группы-владельца */
    dev_t         st_rdev;     /* тип устройства */
                               /* (если это устройство) */
    off_t         st_size;     /* общий размер в байтах */
    blksize_t     st_blksize;  /* размер блока ввода-вывода */
                               /* в файловой системе */
    blkcnt_t      st_blocks;   /* количество выделенных блоков */
    time_t        st_atime;    /* время последнего доступа */
    time_t        st_mtime;    /* время последней модификации */
    time_t        st_ctime;    /* время последнего изменения */
};
```

1.1.3. Выделение памяти в куче

- Выделение

```
#include <stdlib.h>
void *malloc(size_t size);
```

- Выделение с обнулением

```
#include <stdlib.h>
void *calloc(size_t nelem, size_t elsize);
```

- Изменение величины выделенной мамяти (в большую или меньшую сторону)

```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

- Освобождение памяти

```
#include <stdlib.h>
void free(void *ptr);
```

1.2. Использование файлов, отображаемых на память (MMF)

1.3. Файлы, отображаемые на память

- Процесс может запросить ссылку на область памяти ядра, содержащую кэшированный файл

  Т.е. файл кэшируется ОС (page cache), а процесс получает ссылку на область памяти, где файл скэширован.

```
#include <sys/mman.h>
void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
```

  Возвращает указатель на файл в памяти или спициальное значение

```
MAP_FAILED
```

- **addr** - желаемый адрес начала участка памяти, Если **0** - ядро само определяет этот адрес.
- **len** - количество байт файла, которое надо отобразить в память. Должно быть равно размеру файла или меньше его (если надо отобразить не весь файл). Но не больше.
- **prot** - параметры защиты памяти - битовая маска из следующих констант

Значение	Описание
PROT_READ	Доступность на чтение
PROT_WRITE	Доступность на запуск
PROT_EXEC	Доступность на выполнение
PROT_NONE	Данные не должны быть доступны

Защита памяти не установится ниже, чем права, с которыми открыт файл в процессе

- **flag** - атрибуты области, способ отображение в адресное пространство

Значение	Описание
MAP_SHARED	Полученное отображение файла впоследствии будет использоваться и другими процессами, вызвавшими mmap для этого файла с аналогичными значениями параметров, а все изменения, сделанные в отображенном файле, будут сохранены во вторичной памяти
MAP_PRIVATE	Процесс получает отображение файла в свое монопольное распоряжение, но все изменения в нем не могут быть занесены во вторичную память

- **fildes** - файловый дескриптор
- **offset** - смещение относительно начала файла
- Отключение отображения на память

```
#include <sys/mman.h>
int munmap(void *addr, size_t len);
```

1.3.1. Код lesson5:examples/mmf.c

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <stdio.h>
#include <sys/mman.h>

const char *myfile = "index.html";

int main() {
        int fdin;
        struct stat statbuf;
        void *mmf_ptr;
        if ((fdin=open(myfile, O_RDONLY)) < 0) {
                perror(myfile);
                return 1;
        }
        if (fstat(fdin, &statbuf) < 0) {
```

```
                perror(myfile);
                return 1;
        }
        if ((mmf_ptr = mmap(0, statbuf.st_size, PROT_READ, MAP_SHARED, fdin, 0)) == MAP_FAILED) {
                perror("myfile");
                return 1;
        }
        if (write(1,mmf_ptr,statbuf.st_size) != statbuf.st_size) {
                perror("stdout");
                return 1;
        }
        close(fdin);
        munmap(mmf_ptr,statbuf.st_size);
        return 0;
}
```

2. Наводим порядок с проектом

2.1. Запуск для тестирования

1. Локальный

```
$ echo "GET / HTTP/1.1\r\n\r\n" | ./myweb
```

2. Сетевой с терминала

```
$ echo "GET / HTTP/1.1\r\n\r\n" | nc 127.0.0.1 80
```

3. Подключение через браузер

2.2. Примерный код для тестирование в локальном и сетевом режиме

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/mman.h>


#define HTTP_HEADER_LEN 256
#define HTTP_REQUEST_LEN 256
#define HTTP_METHOD_LEN 6
#define HTTP_URI_LEN 100
#define FILE_NAME_LEN 1000
#define LOG_ENTRY_LEN 1000

#define REQ_END 100
#define ERR_NO_URI -100
#define ERR_ENDLESS_URI -101
```

```
char const *index_file = "index.html";

struct http_req {
        char request[HTTP_REQUEST_LEN];
        char method[HTTP_METHOD_LEN];
        char uri[HTTP_URI_LEN];
        char uri_path[HTTP_URI_LEN];
        // uri_params
        // version
        // user_agent
        // server
        // accept
};

int fill_req(char *buf, struct http_req *req) {
        if (strlen(buf) == 2) {
                // пустая строка (\r\n) означает конец запроса
                return REQ_END;
        }
        char *p, *a, *b;
        // Это строка GET-запроса
        p = strstr(buf, "GET");
        if (p == buf) {
                // Строка запроса должна быть вида
                // GET /dir/ HTTP/1.0
                // GET /dir HTTP/1.1
                // GET /test123?r=123 HTTP/1.1
                // и т.п.
                strncpy(req->request, buf, strlen(buf));
                strncpy(req->method, "GET", strlen("GET"));
                a = strchr(buf, '/');
                if ( a != NULL) { // есть запрашиваемый URI
                        b = strchr(a, ' ');
                        if ( b != NULL ) { // конец URI
                                strncpy(req->uri, a, b-a);
                                // пусть пока URI_PATH - то же, что и URI
                                strncpy(req->uri_path, a, b-a);
                        } else {
                                return ERR_ENDLESS_URI;
                                // тогда это что-то не то
                        }
                } else {
                        return ERR_NO_URI;
                        // тогда это что-то не то
                }
        }

        return 0;
```

```
}

int log_req(char* log_path, struct http_req *req) {
        int fd;
        char log_entry[LOG_ENTRY_LEN] = "Sample Log Entry";
        if ((fd = open(log_path, O_WRONLY | O_CREAT | O_APPEND, 0600)) < 0) {
                perror(log_path);
                return 1;
        }
        if (write(fd, log_entry, strlen(log_entry)) != strlen(log_entry)) {
                perror(log_path);
                return 1;
        }
        write(fd, "\n", 1);
        fsync(fd);
        close(fd);
        return 0;
}

int make_resp(char *base_path, struct http_req *req) {
        int fdin;
        struct stat statbuf;
        void *mmf_ptr;
        // определяем на основе запроса, что за файл открыть
        char res_file[FILE_NAME_LEN] = "";
        if (base_path != NULL) {
                strncpy(res_file,base_path,strlen(base_path));
        }
        strcat(res_file,index_file); // вот сюда писать отображение запроса в файловые пути
        // открываем
        if ((fdin=open(res_file, O_RDONLY)) < 0) {
                perror(res_file);
                return 1;
        }
        // размер
        if (fstat(fdin, &statbuf) < 0) {
                perror(res_file);
                return 1;
        }
        // mmf
        if ((mmf_ptr = mmap(0, statbuf.st_size, PROT_READ, MAP_SHARED, fdin, 0)) == MAP_FAILED) {
                perror("myfile");
                return 1;
        }
        // Выводим HTTP-заголовки
        char *http_result = "HTTP/1.1 200 OK\r\n";
        write(1,http_result,strlen(http_result));
        char *http_contype = "Content-Type: text/html\r\n";
        write(1,http_contype,strlen(http_contype));
```

```
        char *header_end = "\r\n";
        write(1,header_end,strlen(header_end));
        // Выводим запрошенный ресурс
        if (write(1,mmf_ptr,statbuf.st_size) != statbuf.st_size) {
                perror("stdout");
                return 1;
        }
        // Подчищаем ресурсы
        close(fdin);
        munmap(mmf_ptr,statbuf.st_size);
        return 0;
}

int main (int argc, char* argv[]) {
        // первый параметр - каталог с контентом
        // второй параметр - каталог для ведения журнала
        char base_path[FILE_NAME_LEN] = "";
        char log_path[FILE_NAME_LEN] = "";
        char const *log_file = "access.log";
        if ( argc > 2 ) { // задан каталог журнализации
                strncpy(base_path, argv[1], strlen(argv[1]));
                strncpy(log_path, argv[2], strlen(argv[2]));
                strcat(log_path,"/");
                strcat(base_path,"/");
        }
        strcat(log_path,log_file);
        char buf[HTTP_HEADER_LEN];
        struct http_req req;
        while(fgets(buf, sizeof(buf),stdin)) {
                int ret = fill_req(buf, &req);
                if (ret == 0)
                        // строка запроса обработана, переходим к следующей
                        continue;
                if (ret == REQ_END )
                        // конец HTTP запроса, вываливаемся на обработку
                        break;
                else
                        // какая-то ошибка
                        printf("Error: %d\n", ret);

        }
        log_req(log_path, &req);
        make_resp(base_path,&req);
}
```

2.3. Инструкции по сборке

```
CFLAGS=-ggdb
TARGET=myweb
```

```
PREFIX=/usr/local
WEBROOT=/srv/myweb
XINETD=/etc/xinetd.d
LOGDIR=/var/log/myweb

.PHONY: all clean install uninstall coreon

all: $(TARGET)

myweb: myweb.c
	cc $(CFLAGS) $^ -o $@

clean:
	-rm -f $(TARGET) access.log

install:
	install $(TARGET) $(PREFIX)/bin
	install $(TARGET)-wrap $(PREFIX)/bin
	[ -d $(WEBROOT) ] || mkdir $(WEBROOT)
	install webroot/* $(WEBROOT)
	install $(TARGET)-xinetd $(XINETD)
	[ -d $(LOGDIR) ] || mkdir $(LOGDIR)

uninstall:
	rm -f $(PREFIX)/bin/$(TARGET)
	-rm -rf $(WEBROOT)
	rm -rf $(XINETD)/$(TARGET)-xinetd

coreon:
	sysctl kernel.core_pattern=core
```

2.4. Настройка сервиса для xinetd

```
service http
{
       socket_type             = stream
       wait                    = no
       user                    = root
       server                  = /usr/local/bin/myweb-wrap
       server_args             = /srv/myweb /var/log/myweb
}
```

2.5. Скрипт-обертка

```
#!/bin/sh
/usr/local/bin/myweb $1 $2 2>> /var/log/myweb/error.log
```