

Федеральное государственное бюджетное образовательное учреждение высшего образования.  
«Национально исследовательский университет «Московский энергетический институт»  
Кафедра ВМСС

Лабораторная работа №3

ОБРАБОТКА ФАЙЛОВ В ОС UNIX

Курс: МУЛЬТИЗАДАЧНЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Группа: А-07м-23

Выполнили: Балашов С.А.,  
Кретов Н.В., Рогов Д.Р.

Проверил: Орлов Д.А.

Москва 2023 г.

7. Напишите свою версию программы CP (назовем ее OURCP), используя описанную выше процедуру COPYFILE и обеспечив с помощью системного вызова TIME замер времени выполнения перезаписи выбранного вами файла.

ourcp.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <time.h>

#define BUFFERSIZE 1024
#define COPYMORE 0644

void oops(char *, char *);
int copyFiles(char *src, char *dest);

int main(int ac, char *av[])
{
    time_t t1, t2;
    if(ac != 3)
    {
        fprintf(stderr, "usage: %s source destination\n", *av);
        exit(1);
    }

    time(&t1);

    char *src = av[1];
    char *dest = av[2];

    if( src[0] != '/' && dest[0] != '/' )//cp1 file1.txt file2.txt
    {
        copyFiles(src, dest);
    }
    else if( src[0] != '/' && dest[0] == '/' )//cp1 file1.txt /dir
    {
        int i;
        for(i=1; i<=strlen(dest); i++)
        {
            dest[(i-1)] = dest[i];
        }
        strcat(dest, "/");
        strcat(dest, src);

        copyFiles(src, dest);
    }
    else
    {
        fprintf(stderr, "usage: cp1 source destination\n");
        exit(1);
    }
    time(&t2);

    fprintf(stderr, "Time: %ld\n", t2 - t1);
}

int copyFiles(char *source, char *destination)
{
    int in_fd, out_fd, n_chars;
    char buf[BUFFERSIZE];

    if( (in_fd=open(source, O_RDONLY)) == -1 )
    {
        oops("Cannot open ", source);
    }

    if( (out_fd=creat(destination, COPYMORE)) == -1 )
    {
```

```

    oops("Cannot creat ", destination);
}

while( (n_chars = read(in_fd, buf, BUFFERSIZE)) > 0 )
{
    if( write(out_fd, buf, n_chars) != n_chars )
    {
        oops("Write error to ", destination);
    }

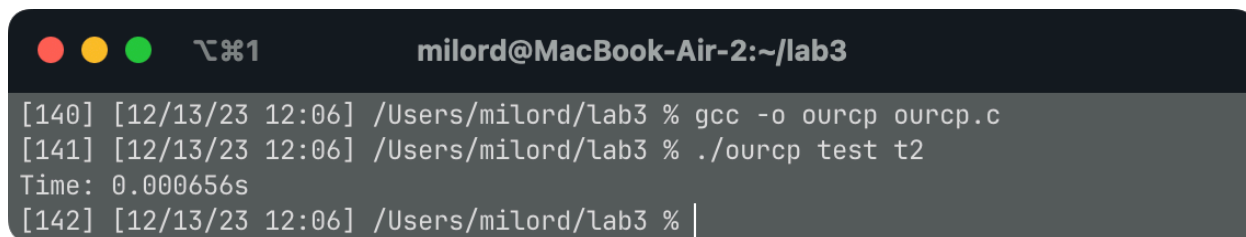
    if( n_chars == -1 )
    {
        oops("Read error from ", source);
    }
}

if( close(in_fd) == -1 || close(out_fd) == -1 )
{
    oops("Error closing files", "");
}

return 1;
}

void oops(char *s1, char *s2)
{
    fprintf(stderr, "Error: %s ", s1);
    perror(s2);
    exit(1);
}

```



A terminal window titled "milord@MacBook-Air-2:~/lab3" showing the following commands and output:

```

[140] [12/13/23 12:06] /Users/milord/lab3 % gcc -o ourcp ourcp.c
[141] [12/13/23 12:06] /Users/milord/lab3 % ./ourcp test t2
Time: 0.000656s
[142] [12/13/23 12:06] /Users/milord/lab3 % |

```

8. Написать две программы, одну, использующую библиотеку F.H, другую, использующую библиотеку L.H . если номер бригады при делении на 3 даёт остаток 2 (варианты 2, 5, 8,...): определите длину вашего файла, разделите его пополам и перепишите под тем же именем. Время полной обработки замерьте;

Таблица 1. Оригинальная библиотека L.h и её модификация

<pre> * * * L.H * * * #include &lt;sys/param.h&gt; #include &lt;fsnt1.h&gt; #include &lt;unistd.h&gt; #define BSIZ 64 #define NOFILE 4 struct f_addr {     long f_blkno; /*обработываемый блок */     int f_valid; /* возвращаемое значение*/     char f_buffer[BSIZ]; /* буфер обмена*/     char f_flag; /* флаг занятости буфера */ }; #define DIRTY 1 /* буфер занят*/ #define EOF (-1) /* конец файла*/ #define ERROR (-2) /* ошибка в-в*/ struct f_addr *f_ptr[NOFILE]; aopen(name, flag) char *name; int flag; {     int fd;     struct f_addr *fp;     fd=open(name,flag);     if(fd&gt;=0){         f_ptr[fd] = malloc(sizeof(struct f_addr));         if(!f_ptr[fd])             return(-1);         fp = f_ptr[fd];         fp-&gt;f_blkno = -1;         fp-&gt;f_flag = fp-&gt;f_valid = 0;     }     return(fd); } aclose(fd) int fd; {     struct f_addr *fp = f_ptr[fd];chkflush(fd);     free(fp);     f_ptr[fd] = 0;     return(close(fd)); } agetb(fd, address) int fd; long address; {     struct f_addr *fp = f_ptr[fd];     int a_offset = address%BSIZ;     long a_blkno = address/BSIZ;     if (fp-&gt;f_blkno != a_blkno)     {         chkflush(fd);         fp-&gt;f_blkno = a_blkno;         lseek(fd, fp-&gt;f_buffer,BSIZ);         fp-&gt;f_valid = read(fd,fp-&gt;f_buffer,BSIZ);         lseek(fd,fp-&gt;f_blkno*BSIZ,0);         fp-&gt;f_valid = read (fd, fp-&gt;f_buffer,BSIZ);         if (fp-&gt;f_valid &lt;= 0)             return(fp-&gt;f_valid - 1);     }     if(a_offset &lt; fp-&gt;f_valid)         return(fp-&gt;f_buffer[a_offset]&amp;0xFF);     else         return(EOF); } asetb (fd, address, value) int fd, value; long address; {     struct f_addr *fp = f_ptr[fd];     int a_offset = address%BSIZ;     long a_blkno = address/BSIZ;     if(fp-&gt;f_blkno != a_blkno)     {         chkflush(fd);         fp-&gt;f_blkno = a_blkno;         lseek(fd, fp-&gt;f_blkno*BSIZ,0);         fp-&gt;f_valid = read(fd, fp-&gt;f_buffer, BSIZ);         if(fp-&gt;f_valid &lt;= 0)             return(fp-&gt;f_valid - 1);     }     fp-&gt;f_buffer[a_offset] = value&amp;0xFF;     fp-&gt;f_flag  = DIRTY;     return(0); } chkflush(fd) int fd; {     struct f_addr *fp = f_ptr[fd];     if(fp-&gt;f_flag &amp; DIRTY)     {         lseek(fd, fp-&gt;f_blkno*BSIZ,0);         return(write(fd, fp-&gt;f_buffer. BSIZ));     } } </pre>	<pre> #include &lt;sys/param.h&gt; #include &lt;fcntl.h&gt; #include &lt;unistd.h&gt; #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #define BSIZ 64 #define NOFILE 4 struct f_addr {     long f_blkno; /*обработываемый блок */     int f_valid; /* возвращаемое значение*/     char f_buffer[BSIZ]; /* буфер обмена*/     char f_flag; /* флаг занятости буфера */ }; #define DIRTY 1 /* буфер занят*/ #define EOF (-1) /* конец файла*/ #define ERROR (-2) /* ошибка в-в*/ struct f_addr *f_ptr[NOFILE]; int chkflush(int fd) {     struct f_addr *fp = f_ptr[fd];     if(fp-&gt;f_flag &amp; DIRTY)     {         lseek(fd, fp-&gt;f_blkno*BSIZ, 0);         return(write(fd, fp-&gt;f_buffer, BSIZ));     }     // return (1); } int aopen(char *name, int flag) {     int fd;     struct f_addr *fp;     fd=open(name,flag);     if (fd&gt;=0)     {         f_ptr[fd] = (struct f_addr*)malloc(sizeof(struct f_addr));         if(!f_ptr[fd])             return(-1);         fp = f_ptr[fd];         fp-&gt;f_blkno = -1;         fp-&gt;f_flag = fp-&gt;f_valid = 0;     }     return(fd); } int aclose(int fd) {     struct f_addr *fp = f_ptr[fd];     chkflush(fd);     free(fp);     f_ptr[fd] = 0;     return(close(fd)); } int agetb(int fd, long address) {     struct f_addr *fp = f_ptr[fd];     int a_offset = address%BSIZ;     long a_blkno = address/BSIZ;     if (fp-&gt;f_blkno != a_blkno)     {         chkflush(fd);         fp-&gt;f_blkno = a_blkno;         lseek(fd, fp-&gt;f_buffer, BSIZ);         fp-&gt;f_valid = read(fd,fp-&gt;f_buffer,BSIZ);         lseek(fd,fp-&gt;f_blkno*BSIZ,0);         fp-&gt;f_valid = read (fd, fp-&gt;f_buffer,BSIZ);         if (fp-&gt;f_valid &lt;= 0)             return(fp-&gt;f_valid - 1);     }     if(a_offset &lt; fp-&gt;f_valid)         return(fp-&gt;f_buffer[a_offset]&amp;0xFF);     else         return(EOF); } int asetb (int fd, long address, int value) {     struct f_addr *fp = f_ptr[fd];     int a_offset = address%BSIZ;     long a_blkno = address/BSIZ;     if(fp-&gt;f_blkno != a_blkno)     {         chkflush(fd);         fp-&gt;f_blkno = a_blkno;         lseek(fd, fp-&gt;f_blkno*BSIZ,0);         fp-&gt;f_valid = read(fd, fp-&gt;f_buffer, BSIZ);         if(fp-&gt;f_valid &lt;= 0)             return(fp-&gt;f_valid - 1);     }     fp-&gt;f_buffer[a_offset] = value&amp;0xFF;     fp-&gt;f_flag = DIRTY;     return(0); } </pre>
---	--

## half1.c

```
#include "./L.h"
#include <sys/stat.h>
#include <time.h>

int k = 0;

void cutfile(char *src);

int main(int ac, char *av[]) {
    time_t t1, t2;
    if(ac != 2)
    {
        fprintf(stderr, "usage: %s source destination\n", *av);
        exit(1);
    }
    k++;

    time(&t1);
    char *src = av[1];
    if (src[0] != '/')
    {
        cutfile(src);
    }
    time(&t2);
    printf("time: %ld\n", t2 - t1);
}

void cutfile(char *src)
{
    int fd;
    if ((fd = aopen(src, O_RDWR)) == -1){
        printf("Invalid path to file\n");
        return;
    }
    k++;
    struct stat fileInfo;
    int status, size;
    status = fstat(fd, &fileInfo);
    // status = stat(src, &buffer);
    if (status == 0)
    {
        size = fileInfo.st_size / 2;
    }
    else {
        printf("Error getting size\n");
        aclose(fd);
        return;
    }

    if (size > 0)
    {
        int *buffer = malloc(size * sizeof(int));
        for (long i = size; i < 2*size; ++i)
        {
            buffer[i - size] = agetb(fd, i);
        }
        for (long i = 0; i < size; ++i)
        {
            asetb(fd, i, buffer[i]);
        }

        free(buffer);
    } else {
        printf("File is empty\n");
    }

    aclose(fd);
    k++;
    truncate(src, size);
}
```

## Таблица 2. Оригинальная библиотека F.h и её модификация

<pre> lib F.H #include &lt;stdio.h&gt;FILE *aopen(name, flag); int flag; switch(flag) { case 0: return(fopen(name, "r")); case 1: return(fopen(name, "w")); case 2: return(fopen(name, "r+")); default:return(NULL); } } int aclose(fp) { return(fclose(fp)); } int agetb(fp, addr) FILE *fp; long addr; { int es; if((es = fseek(fp, addr, SEEK_SET)) != 0) return(es); return (getc (fp)); } int asetb(fp, addr, value) FILE *fp; long addr; char value; { int es; if((es = fseek(fp, addr, SEEK_SET)) != 0) return(es); return (putc (value, fp)); } </pre>	<pre> #include &lt;sys/param.h&gt; #include &lt;fcntl.h&gt; #include &lt;unistd.h&gt; #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; FILE *aopen(char * name, int flag) { switch(flag) { case 0: return(fopen(name, "r")); case 1: return(fopen(name, "w")); case 2: return(fopen(name, "r+")); default:return(NULL); } } int aclose(FILE * fp) { return(fclose(fp)); } int agetb(FILE * fp, long addr) { int es; if((es = fseek(fp, addr, SEEK_SET)) != 0) return (es); return (getc (fp)); } int asetb(FILE *fp, long addr, int value) { int es; if((es = fseek(fp, addr, SEEK_SET)) != 0) return(es); return(putc (value, fp)); } </pre>
--	---

### half2.c

```

#include "./F.h"
#include <sys/stat.h>
#include "time.h"

void cutfile(char *src);

int main(int ac, char *av[]) {
    time_t t1, t2;
    if(ac != 2)
    {
        fprintf(stderr, "usage: %s source destination\n", *av);
        exit(1);
    }

    time(&t1);
    char *src = av[1];
    if (src[0] != '/')
    {
        cutfile(src);
    }
    time(&t2);
    printf("time: %ld\n", t2 - t1);
}

void cutfile(char *src)
{
    FILE *fd;
    if ((fd = aopen(src, 2)) == NULL){
        printf("Invalid path to file\n");
        return;
    }
    struct stat fileInfo;
    int status, size;
    status = stat(src, &fileInfo);
    if (status == 0)
    {
        size = fileInfo.st_size / 2;
    }
    else {
        printf("Error getting size\n");
        aclose(fd);
        return;
    }
}

```

```

}

if (size > 0) {
    int *buffer = malloc(size * sizeof(int));
    for (long i = size; i < 2*size; ++i)
    {
        buffer[i - size] = agetb(fd, i);
    }
    for (long i = 0; i < size; ++i)
    {
        asetb(fd, i, buffer[i]);
    }

    free(buffer);
} else {
    printf("File is empty\n");
}

fclose(fd);
truncate(src, size);
}

```

9. Откомпилируйте написанные программы, устранив синтаксические и семантические ошибки.

```

milord@MacBook-Air-2:~/lab3
[58] [12/08/23 16:49] /Users/milord/lab3 % gcc -o half1 half1.c
In file included from half1.c:1:
././L.h:8:9: warning: 'NOFILE' macro redefined [-Wmacro-redefine
d]
#define NOFILE 4
      ^
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/
sys/param.h:101:9: note: previous definition is here
#define NOFILE      256          /* default max open file
s per process */
      ^
In file included from half1.c:1:
././L.h:34:1: warning: non-void function does not return a value
in all control paths [-Wreturn-type]
}
^
././L.h:76:13: warning: incompatible pointer to integer conversi
on passing 'char[64]' to parameter of type 'off_t' (aka 'long lo
ng') [-Wint-conversion]
        lseek(fd, fp->f_buffer, BSIZE); // BSIZE
              ^~~~~~
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/
unistd.h:465:24: note: passing argument to parameter here
off_t    lseek(int, off_t, int);
              ^
3 warnings generated.
[59] [12/08/23 16:50] /Users/milord/lab3 % |

milord@MacBook-Air-2:~/lab3
[33] [12/11/23 17:42] /Users/milord/lab3 % gcc -o half2 half2.c
[34] [12/11/23 17:42] /Users/milord/lab3 % gcc -o ourcp ourcp.c
[35] [12/11/23 17:42] /Users/milord/lab3 % |

```

10. Используя GDB, проведите отладку программ. Продемонстрируйте работу команд break, watch, print.

```
lldb ./half1 t1
[72] [12/08/23 17:25] /Users/milord/lab3 % lldb ./half1 t1
(lldb) target create "./half1"
Current executable set to '/Users/milord/lab3/half1' (arm64).
(lldb) settings set -- target.run-args "t1"
(lldb) b cutfile
Breakpoint 1: where = half1`cutfile, address = 0x0000000100003ce8
(lldb) print k
(void *) 0x0000000000000000
(lldb) r
Process 18949 launched: '/Users/milord/lab3/half1' (arm64)
Process 18949 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100003ce8 half1`cutfile
half1`cutfile:
-> 0x100003ce8 <+0>: sub    sp, sp, #0xd0
    0x100003cec <+4>: stp    x29, x30, [sp, #0xc0]
    0x100003cf0 <+8>: add    x29, sp, #0xc0
    0x100003cf4 <+12>: stur   x0, [x29, #-0x8]
Target 0: (half1) stopped.
(lldb) print k
(void *) 0x0000000000000001
(lldb) w s e -- 0x100003cf0
Watchpoint created: Watchpoint 1: addr = 0x100003cf0 size = 8 state = enabled type = w
    watchpoint spec = '0x100003cf0'
    new value: -567590302374493187
(lldb) c
Process 18949 resuming
time: 30
Process 18949 exited with status = 0 (0x00000000)
(lldb) |
```

11. Сравнением времен выполнения различных программ, написанных в п.8, определите значения O, R, N. Они не всегда будут достоверны. Объясните причину.

Таблица 1. Времена выполнения программ

Программа	half1.c (поблоково)	half2.c (побайтово)
Файл 16 Кб	0.001856	0.050604
Файл 64 Кб	0.006504	0.142328
Файл 1 Мб	0.059410	1.720701

S1 = 16 Кбайт = 16384 байт  
S2 = 64 Кбайт = 65536 байт  
S3 = 1 Мбайт = 1048576 байт  
N = 64



$$\begin{aligned}
T_{\text{побайт1}} &= 0.050604 \text{ с} = 2 * S1 * (O + R) \\
T_{\text{побайт2}} &= 0.142328 \text{ с} = 2 * S2 * (O + R) \\
T_{\text{побайт3}} &= 1.720701 \text{ с} = 2 * S3 * (O + R) \\
T_{\text{поблок1}} &= 0.001856 \text{ с} = 2 * S1 * (O + R_n) / N \\
T_{\text{поблок2}} &= 0.006504 \text{ с} = 2 * S2 * (O + R_n) / N \\
T_{\text{поблок3}} &= 0.05941 \text{ с} = 2 * S3 * (O + R_n) / N \\
R_n &= N * R
\end{aligned}$$

$$R1 = (N * T_{\text{поблок1}} - T_{\text{побайт1}}) / (2 * S1 * (N - 1)) = (64 * 0.001856 - 0.050604) / (2 * 16384 * (64 - 1)) = \mathbf{3.30 * 10^{-8}}$$

$$O1 = T_{\text{побайт1}} / (2 * S1) - R1 = 0.050604 / (2 * 16384) - 3.3 * 10^{-8} = \mathbf{1.5 * 10^{-6}}$$

$$R2 = (N * T_{\text{поблок2}} - T_{\text{побайт2}}) / (2 * S2 * (N - 1)) = (64 * 0.006504 - 0.142328) / (2 * 65536 * (64 - 1)) = \mathbf{3.32 * 10^{-8}}$$

$$O2 = T_{\text{побайт2}} / (2 * S2) - R2 = 0.142328 / (2 * 65536) - 3.32 * 10^{-8} = \mathbf{1.05 * 10^{-6}}$$

$$R3 = (N * T_{\text{поблок3}} - T_{\text{побайт3}}) / (2 * S3 * (N - 1)) = (64 * 0.05941 - 1.720701) / (2 * 1048576 * (64 - 1)) = \mathbf{1.58 * 10^{-8}}$$

$$O3 = T_{\text{побайт3}} / (2 * S3) - R3 = 1.720701 / (2 * 1048576) - 1.58 * 10^{-8} = \mathbf{8.05 * 10^{-7}}$$

12. Провести две серии экспериментов. Для их проведения приготовить пять исходных файлов размерами 16K, 64K, 256K, 1M, 4M. Первую серию из 5 экспериментов провести для библиотеки F.H, решая задачу п.8 для пяти файлов. Времена выполнения занести в таблицу TF. Вторую серию из 25 экспериментов провести для библиотеки L.H, решая задачу п.8 для пяти файлов, при этом обрабатывая каждый файл блоками пяти разных размеров. Времена выполнения занести в таблицу TL. Провести сравнение времён в таблицах TF и TL. Выбрать лучшие и худшие времена, а также лучший и худший режим (символьный, блочный) для каждого из пяти файлов. Сделать выводы.

Таблица TF

Размер	Время
16K	0.049173
64K	0.137603
256K	0.339893
1M	1.709379
4M	4.985803

Таблица TL

Размер	Время (блок 32)	Время (блок 64)	Время (блок 128)	Время (блок 256)	Время (блок 512)
16K	0.004927	0.002470	0.001779	0.001317	0.000802
64K	0.013098	0.006795	0.004506	0.002743	0.001925
256K	0.040859	0.024148	0.015000	0.010222	0.006977
1M	0.108674	0.071326	0.046556	0.031390	0.023532
4M	0.323508	0.194921	0.117484	0.083414	0.062689

С увеличением размера блока при блочном режиме уменьшается время выполнения для каждого из файлов. Каждое из времен блочного режима лучше времен символьного режима.

Лучшим режимом в данном случае является блочный