Поправителен изпит, Задачи, част II, курс 'Операционни системи', СУ, ФМИ, 25.08.2016 г.

Име: _____ ФН: _____ Спец.: __ Курс: __ Група: __

Задача	3	4	Общо
получени точки			
максимум точки	20	20	40

Задача 1.

Напишете програма на C, която да работи като обвивка на командата `sort` – тоест, вашата програма изпълнява `sort`, като всички параметри подадени на командния ред, да се предават на `sort`. Изхода за грешки по време на изпълнението да отива във файл с име `serror.txt`.

Задача 2.

Напишете програма на C, която приема три параметъра, имена на двоични файлове. Примерно изпълнение:

```
./main patch.bin f1.bin f2.bin
```

Файловете `f1.bin` и `f2.bin` се третират като двойчни файлове, състоящи се от байтове (`uint8_t`). Файлът `patch.bin` е двоичен файл, състоящ се от наредени тройки от следните елементи (и техните типове):

- *отместване* `uint16_t`
- *оригиналенбайт* `uint8_t`
- *новбайт* `uint8_t`

Програмата да създава файла `f2.bin` като копие на файла `f1.bin`, но с отразени промени на базата на файла `patch.bin`, при следният алгоритъм:

- за всяка наредена тройка от `patch.bin`, ако на съответното *отместване* (в байтове), спрямо началото на файла е записан байта *оригиналенбайт*, в изходния файл се записва *новбайт*. Ако не е записан такъв *оригиналенбайт* или такова *отместване* не съществува, програмата да прекратява изпълнението си по подходящ начин
- всички останали байтове се копират директно

Забележка: Наредените тройки във файла patch.bin да се обработват последователно.

Примерен `f1.bin`:

```
00000000: f5c4 b159 cc80 e2ef c1c7 c99a 2fb0 0d8c  ...Y......../...
00000010: 3c83 6fed 6b46 09d2 90df cf1e 9a3c 1f05  <.o.kF.......<..
00000020: 05f9 4c29 fd58 a5f1 cb7b c9d0 b234 2398  ..L).X...{...4#.
00000030: 35af 6be6 5a71 b23a 0e8d 08de def2 214c  5.k.Zq.:......!L
```

Примерен `patch.bin`:

```
00000000: 0200 b159 3000 35af                      ...Y0.5.
```

Примерен `f2.bin`:

```
00000000: f5c4 5959 cc80 e2ef c1c7 c99a 2fb0 0d8c  ..YY......../...
00000010: 3c83 6fed 6b46 09d2 90df cf1e 9a3c 1f05  <.o.kF.......<..
00000020: 05f9 4c29 fd58 a5f1 cb7b c9d0 b234 2398  ..L).X...{...4#.
00000030: afaf 6be6 5a71 b23a 0e8d 08de def2 214c  ..k.Zq.:......!L
```

Забележки:

- Примерни системни извиквания:

  `open()` `close()` `read()` `write()` `lseek()` `scanf()` `pipe()` `dup2()` `fork()` `wait()` `waitpid()` `exec()` `execv()`

- Препоръчителни флагове на компилатора: `-std=c99 -Wall -Wpedantic -Wextra`
- Обърнете внимание на коментарите, именуването на променливи и подреждането на кода

Задача 1. Примерно решение

```
#include <err.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
    int f = open("serror.txt", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    dup2(f, 2);

    if (execvp("sort", argv) == -1) {
        err(1, "exec sort");
    }
}
```

Задача 2. Примерно решение

```
#include <stdlib.h>
#include <err.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdint.h>

#include <stdio.h>

int main (int argc, char* argv[])
{
    if (argc != 4) {
        errx(1, "Invalid number of arguments. Usage: %s <patch.bin> <f1.bin> <f2.bin>", argv[0]);
    }

    int pf, f1, f2;

    pf = open(argv[1], O_RDONLY);
    if (pf == -1) {
        err(1, "%s", argv[1]);
    }

    f1 = open(argv[2], O_RDONLY);
    if (f1 == -1) {
        int saved_errno = errno;
        close(pf);
        errno = saved_errno;
        err(1, "%s", argv[2]);
    }

    f2 = open(argv[3], O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (f2 == -1) {
        int saved_errno = errno;
        close(pf);
        close(f1);
        errno = saved_errno;
        err(1, "%s", argv[3]);
    }

    uint8_t buf[4096];
    ssize_t read_size = 0;
```

```
    while ((read_size = read(f1, buf, sizeof(buf))) > 0) {
        if (write(f2, buf, read_size) != read_size) {
            err(1, "error writing to %s", argv[2]);
        }
    }

    close(f1);
    lseek(f2, 0, SEEK_SET);

    struct {
        uint16_t offset;
        uint8_t orgbyte;
        uint8_t newbyte;
    } element;

    uint8_t byte = 0;

    while (read(pf, &element, sizeof(element)) == sizeof(element)) {
        printf("%04x %02x %02x\n", element.offset, element.orgbyte, element.newbyte);

        if (lseek(f2, element.offset, SEEK_SET) < 0 ) {
            int saved_errno = errno;
            close(f2);
            errno = saved_errno;
            err(1, "lseek in %s to %04x failed", argv[2], element.offset);
        }

        if (read(f2, &byte, 1) != 1) {
            int saved_errno = errno;
            close(f2);
            errno = saved_errno;
            err(1, "read from %s at %04x failed", argv[2], element.offset);
        }

        if (byte != element.orgbyte) {
            int saved_errno = errno;
            close(f2);
            errno = saved_errno;
            err(1, "byte in %s at %04x is different from %02x",
                argv[2], element.offset, element.orgbyte);
        }

        lseek(f2, -1, SEEK_CUR);

        if (write(f2, &element.newbyte, 1) != 1) {
            int saved_errno = errno;
            close(f2);
            errno = saved_errno;
            err(1, "write in %s at %04x of %02x failed",
                argv[2], element.offset, element.newbyte);
        }
    }

    close(f2);
    close(pf);

    exit(0);
}
```