Зад. 1
Реализирайте команда cp, работеща с два аргумента, подадени като входни параметри.

Зад. 2
Реализирайте команда cat, работеща с произволен брой подадени входни параметри.

Зад. 3
Реализирайте команда cp, работеща с произволен брой подадени входни параметри.

Зад. 4
Копирайте файл /etc/passwd в текущата ви работна директория и променете разделителят на копирания файл от ":", на "?"

Зад. 5
Напишете програма, която приема точно 2 аргумента. Първият може да бъде само --min, --max или --print. Вторият аргумент е двоичен файл, в който има записани цели неотрицателни двубайтови числа. Ако първият аргумент е:

- --min - програмата отпечатва кое е най-малкото число в двоичния файл
- --max - програмата отпечатва кое е най-голямото число в двоичния файл
- --print - програмата отпечатва на нов ред всяко число.

Зад. 6
Модифицирайте предната програма, така че да намерите най-големия байт.

Зад. 7
Да се напише програма на C, която получава като параметър команда (без параметри) и при успешното ѝ изпълнение, извежда на стандартния изход името на командата.

Зад. 8
Да се напише програма на C, която получава като параметри три команди (без параметри), изпълнява ги последователно, като изчаква края на всяка и извежда на стандартния изход номера на завършилия процес, както и неговия код на завършване.

Зад. 9
Да се напише програма на C, която получава като параметър име на файл. Създава процес син, който записва стринга foobar във файла (ако не съществува, го създава, в противен случай го занулява), след което процеса родител прочита записаното във файла съдържание и го извежда на стандартния изход, добавяйки по един интервал между всеки два символа.

Зад. 10
Да се напише програма на C, която която създава файл в текущата директория и генерира два процеса, които записват низовете foo и bar в създадения файл. Програмата записва низовете последователно, като първия е foo.

Зад. 11
Да се напише програма на C, която получава като параметри от команден ред две команди (без параметри). Изпълнява първата. Ако тя е завършила успешно изпълнява втората. Ако не, завършва с код 42.

Зад. 12
Да се напише програма на C, която изпълнява последователно подадените ѝ като параметри команди, като реализира следната функционалност постъпково:

- main cmd1 ... cmdN Изпълнява всяка от командите в отделен дъщерен процес
- ... при което се запазва броя на изпълнените команди, които са дали грешка и броя на завършилите успешно.

Зад. 13
Да се напише програма на C, която получава като параметри от команден ред две команди (без параметри) и име на файл в текущата директория. Ако файлът не съществува, го създава. Програмата изпълнява командите последователно, по реда на подаването им. Ако първата команда завърши успешно, програмата добавя нейното име към съдържанието на файла, подаден като команден параметър. Ако командата завърши неуспешно, програмата уведомява потребителя чрез подходящо съобщение.

Зад. 14
Да се напише програма на C, която получава като командни параметри две команди (без параметри). Изпълнява ги едновременно и извежда на стандартния изход номера на процеса на първата завършила успешно. Ако нито една не завърши успешно извежда -1.

Зад. 15
Напишете програма на C, която приема параметър - име на (двоичен) файл с байтове. Програмата трябва да сортира файла.

Зад. 16

Напишете програма на C, която да работи подобно на командата cat, реализирайки само следната функционалност:

- общ вид на изпълнение: ./main [OPTION] [FILE]...
- ако е подаден като първи параметър -n, то той да се третира като опция, което кара програмата ви да номерира (глобално) всеки изходен ред (започвайки от 1)
- програмата извежда на STDOUT
- ако няма подадени параметри (имена на файлове), програмата чете от STDIN
- ако има подадени параметри – файлове, програмата последователно ги извежда
- ако някой от параметрите е тире (-), програмата да го третира като специално име за STDIN.

Примерно извикване:

```
$ cat a.txt
        a1
        a2
$ cat b.txt
        b1
        b2
        b3
$ echo -e "s1\ns2" | ./main -n a.txt - b.txt
        1 a1
        2 a2
        3 s1
        4 s2
        5 b1
        6 b2
        7 b3
```

Зад. 17

Напишете програма на C, която да работи подобно на командата cat, реализирайки само следната функционалност:

- програмата извежда на STDOUT
- ако няма подадени параметри, програмата чете от STDIN
- ако има подадени параметри – файлове, програмата последователно ги извежда
- ако някой от параметрите започва с тире (-), програмата да го третира като специално име за STDIN.

Примерно извикване: $ ./main f - g
извежда съдържанието на файла f, после STDIN, след това съдържанието на файла g.

Зад. 18

Напишете програма на C, която да работи подобно на командата tr, реализирайки само следната функционалност:

- програмата чете от стандартния вход и пише на стандартния изход
- общ вид на изпълнение: ./main [OPTION] SET1 [SET2]
- OPTION би могъл да бъде или -d, или -s, или да липсва
- SET1 и SET2 са низове, в които знаците нямат специално значение, т.е. всеки знак “означава” съответния знак. SET2, ако е необходим, трябва да е същата дължина като SET1
- ако е подаден като първи параметър -d, програмата трие от входа всяко срещане на знак μ от SET1; SET2 не е необходим
- ако е подаден като първи параметър -s, програмата заменя от входа всяка поредица от повторения знак μ от SET1 с еднократен знак μ; SET2 не е необходим
- в останалите случаи програмата заменя от входа всеки знак μ от SET1 със съответстващият му позиционно знак ν от SET2.

Примерно извикване: $ echo asdf | ./main -d ’d123’ | ./main ’sm’ ’fa’ | ./main -s ’f’
af

Зад. 19

Напишете програма на C, която да работи подобно на командата cut, реализирайки само следната функционалност:

- програмата трябва да чете текст от стандартния вход и да извежда избраната част от всеки ред на стандартния изход
- ако първият параметър на програмата е низът -c, тогава вторият параметър е или едноцифрено число (от 1 до 9), или две едноцифрени числа N и M ($N \leq M$), разделени с тире (напр. 3-5). В този случай програмата трябва да изведе избраният/избраните символи от реда: или само символа на указаната позиция, или няколко последователни символи на посочените позиции
- ако първият параметър на програмата е низът -d, тогава вторият параметър е низ, от който е важен само първият символ; той се използва като разделител между полета на реда. Третият параметър трябва да бъде низът -f, а четвъртият - или едноцифрено число (от 1 до 9), или две едноцифрени числа N и M ($N \leq M$), разделени с тире (напр. 3-5). В този случай програмата трябва да разглежда реда като съставен от няколко полета (може би

празни), разделени от указания символ (първият символ от низа, следващ параметъра -d), и да изведе или само указаното поле, или няколко последователни полета на указаните позиции, разделени от същия разделител
- ако някой ред няма достатъчно символи/полета, за него програмата трябва да изведе каквото (докъдето) е възможно (или дори празен ред).

Зад. 20
Напишете програма на C приемаща параметър – име на (двоичен) файл с uint32_t числа. Програмата трябва да сортира файла. Ограничения:
- Числата биха могли да са максимум 100 000 000 на брой
- Програмата трябва да работи на машина със същия endianness, както машината, която е създала файла
- Програмата трябва да работи на машина с 256 MB RAM и 8 GB свободно дисково пространство.

Зад. 21
Напишете програма на C, която приема два параметъра – имена на файлове:
- примерно извикване: ./main input.bin output.bin
- файловете input.bin и output.bin се третират като двоични файлове, състоящи се от uint32_t числа
- файлът input.bin може да съдържа максимум 4194304 числа
- файлът output.bin трябва да бъде създаден от програмата и да съдържа числата от input.bin, сортирани във възходящ ред
- endianness-ът на машината, създала файла input.bin е същият, като на текущата машина
- ограничения на ресурси: програмата трябва да работи с употреба на максимум 9 MB RAM и 64 MB дисково пространство.

Зад. 22
Напишете програма на C, която приема два параметъра – имена на файлове:
- примерно извикване: ./main input.bin output.bin
- файловете input.bin и output.bin се третират като двоични файлове, състоящи се от uint16_t числа
- файлът input.bin може да съдържа максимум 65535 числа
- файлът output.bin трябва да бъде създаден от програмата и да съдържа числата от input.bin, сортирани във възходящ ред
- endianness-ът на машината, създала файла input.bin е същият, като на текущата машина
- ограничения на ресурси: програмата трябва да работи с употреба на максимум 256 KB RAM и 2 MB дисково пространство.

Зад. 23
Напишете програма на C, която приема четири параметъра – имена на двоични файлове.
Примерно извикване: $ ./main f1.dat f1.idx f2.dat f2.idx
Първите два (f1.dat и f1.idx) и вторите два (f2.dat и f2.idx) файла са входен и изходен комплект със следния смисъл:
- DAT-файловете (f1.dat и f2.dat) представляват двоични файлове, състоящи се от байтове (uint8_t)
- IDX-файловете представляват двоични файлове, състоящи се от наредени тройки от следните елементи (и техните типове), които дефинират поредици от байтове (низове) от съответния DAT файл:
  - отместване uint16_t – показва позицията на първия байт от даден низ спрямо началото на файла
  - дължина uint8_t – показва дължината на низа
  - запазен uint8_t – не се използва.

Първата двойка файлове (f1.dat и f1.idx) съществува, а втората трябва да бъде създадена от програмата по следния начин:
- трябва да се копират само низовете (поредици от байтове) от входния комплект, които започват с главна латинска буква (A - 0x41, Z - 0x5A)
- ако файловете са неконсистентни по някакъв начин, програмата да прекратява изпълнението си по подходящ начин.

Приемаме, че DAT файлът съдържа текстови данни на латински с ASCII кодова таблица (един байт за буква).

Зад. 24
Напишете програма на C, която приема три параметъра – имена на двоични файлове.
Примерно извикване: $ ./main f1.bin f2.bin patch.bin
Файловете f1.bin и f2.bin се третират като двоични файлове, състоящи се от байтове (uint8_t).
Файлът f1.bin е "оригиналният" файл, а f2.bin е негово копие, което е било модифицирано по някакъв начин (извън обхвата на тази задача). Файлът patch.bin е двоичен файл, състоящ се от наредени тройки от следните елементи (и техните типове):
- отместване (uint16_t) – спрямо началото на f1.bin/f2.bin
- оригинален байт (uint8_t) – на тази позиция в f1.bin
- нов байт (uint8_t) – на тази позиция в f2.bin.

Вашата програма да създава файла patch.bin, на базата на съществуващите файлове f1.bin и f2.bin, като описва вътре само разликите между двата файла. Ако дадено отместване съществува само в единия от файловете f1.bin/f2.bin, програмата да прекратява изпълнението си по подходящ начин.

Зад. 25
Напишете програма на C, която приема три параметъра, имена на двоични файлове.
Примерно извикване: $ ./main patch.bin f1.bin f2.bin
Файловете f1.bin и f2.bin се третират като двоични 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
- оригинален байт uint8_t
- нов байт 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 следния алгоритъм:

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

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

Зад. 26
Инженерите от съседната лаборатория ползват специализиран хардуер и софтуер за прехвърляне на данни по радио, но за съжаление имат два проблема:

- в радио частта: дълги поредици битове само 0 или само 1 чупят преноса
- в софтуерната част: софтуерът, който ползват, може да прехвърля само файлове с четен брой байтове дължина.

Помогнете на колегите си, като напишете програма на C, която решава тези проблеми, като подготвя файлове за прехвърляне. Програмата трябва да приема два задължителни позиционни аргумента – имена на файлове.
Примерно извикване: $ ./main input.bin output.bin
Програмата чете данни от input.bin и записва резултат след обработка в output.bin. Програмата трябва да работи като encoder, който имплементира вариант на Manchester code, т.е.:

- за всеки входен бит 1 извежда битовете 10, и
- за всеки входен бит 0 извежда битовете 01.

Например, следните 8 бита вход 1011 0110 == 0xB6
по описаният алгоритъм дават следните 16 бита изход 1001 1010 0110 1001 == 0x9A69

Зад. 27
Вашите колеги от съседната лаборатория са написали програма на C, която може да обработва подаден входен двоичен файл и на негова база генерира изходен двоичен файл.
Програмата работи като encoder, който имплементира вариант на Manchester code, т.е.:

- за всеки входен бит 1 извежда битовете 10, и
- за всеки входен бит 0 извежда битовете 01.

Например следните 8 бита вход 1011 0110 == 0xB6
по описания алгоритъм дават следните 16 бита изход 1001 1010 0110 1001 == 0x9A69
Напишете програма на C, която извършва обратния процес, т.е., декодира файлове, създадени от горната програма.
Примерно извикване: $ ./main input.bin output.bin

Зад. 28
Двоичните файлове f1 и f2 съдържат 32 битови числа без знак (uint32_t). Файлът f1 е съдържа n двойки числа, нека i-тата двойка е $< x_i , y_i >$. Напишете програма на C, която извлича интервалите с начало x_i и дължина y_i от файла f2 и ги записва залепени в изходен файл f3.
Пример:

- 1 съдържа 4 числа (2 двойки): 30000, 20, 19000, 10
- програмата записва в f3 две поредици 32-битови числа, взети от f2 както следва:
- най-напред се записват числата, които са на позиции 30000, 30001, 30002, ... 30019
- след тях се записват числата от позиции 19000, 19001, ... 19009.

Зад. 29
Напишете програма на C, която приема шест задължителни позиционни параметъра – имена на файлове.
Примерно извикване: $ ./main affix postfix prefix infix suffix crucifixus

Всички файлове започват с хедър с фиксирана дължина от 16 байта. Пети и шести (спрямо Z^+) байт от хедъра дефинират uint16_t число count, което описва броя на елементите във файла. Файловете affix и infix се състоят от елементи от тип uint16_t, файловете prefix и crucifixus – от елементи от тип uint8_t, postfix – от uint32_t, а suffix – от uint64_t.
Интервал наричаме наредена двойка числа, която дефинира номер (спрямо Z) на начален елемент и брой елементи от даден файл. Комплект наричаме наредена четворка от интервали, които позиционно се отнасят спрямо файловете {post,pre,in,suf}fix.
Елементите на файла affix дефинират серия от комплекти, на чиято база програмата трябва да генерира изходния файл crucifixus.

Зад. 30

Инженерите от съседната лаборатория работят с комплекти SCL/SDL файлове, напр. input.scl/input.sdl.
В SCL файла са записани нива на сигнали (ниско 0 или високо 1), т.е., файлът се третира като състоящ се от битове. В SDL файла са записани данни от тип uint16_t, като всеки елемент съответства позиционно на даден бит от SCL файла.
Помогнете на колегите си, като напишете програма на C, която да бъде удобен инструмент за изваждане в нов файл само на тези SDL елементи, които са имали високо ниво в SCL файла, запазвайки наредбата им.

Зад. 31
Напишете програма на C, която приема два позиционни параметъра – имена на файлове.
Примерно извикване: $ ./main input.bin output.h
Файлът input.bin е двоичен файл с елементи uint16_t числа, създаден на little-endian машина.
Вашата програма трябва да генерира C хедър файл, дефиниращ масив с име arr, който:

- съдържа всички елементи от входния файл
- няма указана големина
- не позволява промяна на данните.

Генерираният хедър файл трябва да:

- съдържа и uint32_t променлива arrN, която указва големината на масива
- бъде валиден и да може да се #include-ва без проблеми от C файлове, очакващи да "виждат" arr и arrN.

За да е валиден един входен файл, той трябва да съдържа не повече от 524288 елемента.

Зад. 32
В приложната статистика често се използват следните описателни статистики за дадена извадка:

- средна стойност $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
- дисперсия $D = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$,

  където $\{x_1, x_2, \ldots, x_N\}$ са стойностите на извадката, а N е техния брой.

Напишете програма на C, която приема задължителен параметър – име на двоичен файл. Файлът съдържа информация за потребители, които са влизали в системата, и се състои от наредени петорки от следните елементи и техните типове:

- уникален потребителски идентификатор (UID) uint32_t
- запазено1 uint16_t – не се използва в задачата
- запазено2 uint16_t – не се използва в задачата
- време1 uint32_t – момент на започване на сесията (Unix time)
- време2 uint32_t – момент на завършване на сесията (Unix time).

За потребители, които са имали сесии, квадратът на продължителността на които е по-голям от дисперсията D на продължителността на всички сесии във файла, програмата трябва да изведе на STDOUT потребителският им идентификатор и продължителността на най-дългата им сесия.
Можете да разчитате на това, че във файла ще има не повече от 16384 записа и че в тях ще се срещат не повече от 2048 различни потребителски идентификатора.

Зад. 33
Напишете програма на C, която приема три параметъра – имена на двоични файлове.
Примерно извикване: $ ./main patch.bin f1.bin f2.bin
Файловете patch.bin и f1.bin съществуват, и на тяхна база програмата трябва да създаде f2.bin.
Файлът patch.bin се състои от две секции – 16 байтов хедър и данни. На базата на хедъра програмата трябва да може да интерпретира съдържанието на файла. Структурата на хедъра е:

- uint32_t, magic – магическа стойност 0xEFBEADDE, която дефинира, че файлът следва тази спецификация
- uint8_t, header version – версия на хедъра, с единствена допустима стойност за момента 0x01, която дефинира останалите байтове от хедъра както следва:
  - uint8_t, data version – версия (описание) на използваните структури в секцията за данни на файла
  - uint16_t, count – брой записи в секцията за данни
  - uint32_t, reserved 1 – не се използва
  - uint32_t, reserved 2 – не се използва.

Възможни структури в секцията за данни на файла спрямо data version:

- при версия 0x00
  - uint16_t, offset
  - uint8_t, original byte
  - int8_t, new byte
- при версия 0x01
  - uint32_t, offset
  - uint16_t, original word
  - uint16_t, new word
- и при двете описани версии offset е отместване в брой елементи спрямо началото на файла
- Двоичните файлове f1.bin и f2.bin се третират като състоящи се от елементи спрямо data version в patch.bin.

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

- за всяка наредена тройка от секцията за данни на patch.bin, ако на съответният offset в оригиналния файл f1.bin е записан елементът original byte/word, в изходният файл се записва new byte/word. Ако не е записан такъв елемент или той не съществува, програмата да прекратява изпълнението си по подходящ начин
- всички останали елементи се копират директно.

Наредените тройки в секцията за данни на файла patch.bin да се обработват последователно.

Зад. 34

Вашите колеги от съседната лаборатория работят със специализирана система Hoge, която съхранява данните си в binary файлове. За съжаление обаче им се налага да работят с две различни версии на системата (стара и нова) и разбира се, те ползват различни файлови формати.

Вашата задача е да напишете програма на C (./main list.bin data.bin out.bin), която конвертира файлове с данни от стария (list.bin + data.bin) в новия (out.bin) формат.

Всички файлове се състоят от две секции – 8 байтов хедър и данни. Структурата на хедъра е:

- uint16_t, magic – магическа стойност 0x5A4D, която дефинира, че файлът е от системата Hoge и следва тази спецификация
- uint16_t, filetype – дефинира какъв тип файл е това, с допустими стойности 1 за list.bin, 2 за data.bin и 3 за out.bin
- uint32_t, count – дефинира броя на елементите в секцията с данни на файла.

Секцията за данни на всички файлове съдържа елементи – цели числа без знак, от съответните типове:

- list.bin – uint16_t
- data.bin – uint32_t
- out.bin – uint64_t.

Във файлове data.bin и out.bin елементи, чиято стойност е 0, са валидни елементи, но се игнорират от системата Hoge (тяхното наличие във файла не пречи на работата на системата).

Всеки елемент в секцията за данни на list.bin има две характеристики – позиция и стойност, като позиция е отместването на съответния елемент, докато стойност е неговата стойност. Тези две числа семантично дефинират отмествания във файловете с данни:

- позиция дефинира отместване в data.bin
- стойност дефинира отместване в out.bin.

На базата на тези числови характеристики елементите на list.bin дефинират правила от вида: “елементът на отместване позиция в data.bin трябва да отиде на отместване стойност в out.bin”.

Всички отмествания са спрямо N_0 в брой елементи.

Програмата трябва да интерпретира по ред дефинираните правила и на тяхна база да генерира изходния файл.

Не е гарантирано, че разполагате с достатъчно памет, в която да заредите всички елементи на който и да от файловете.

Зад. 35

Напишете програма на C, която приема два позиционни аргумента – имена на двоични файлове.

Примерно извикване: ./main data.bin comparator.bin

Файлът data.bin се състои от две секции – 8 байтов хедър и данни. Структурата на хедъра е:

- uint32_t, magic – магическа стойност 0x21796F4A, която дефинира, че файлът следва тази спецификация
- uint32_t, count – описва броя на елементите в секцията с данни.

Секцията за данни се състои от елементи – uint64_t числа.

Файлът comparator.bin се състои от две секции – 16 байтов хедър и данни. Структурата на хедъра е:

- uint32_t, magic1 – магическа стойност 0xAFBC7A37
- uint16_t, magic2 – магическа стойност 0x1C27
- комбинацията от горните две magic числа дефинира, че файлът следва тази спецификация;
- uint16_t, reserved – не се използва
- uint64_t, count – описва броя на елементите в секциата с данни.

Секцията за данни се състои от елементи – наредени 6-торки:

- uint16_t, type – възможни стойности: 0 или 1
- 3 бр. uint16_t, reserved – възможни стойности за всяко: 0
- uint32_t, offset1
- uint32_t, offset2.

Двете числа offset дефинират отместване (спрямо N_0) в брой елементи за data.bin; type дефинира операция за сравнение:

- 0: “по-малко”
- 1: “по-голямо”.

Елементите в comparator.bin дефинират правила от вида:

- “елементът на offset1” трябва да е “по-малък” от “елементът на offset2”
- “елементът на offset1” трябва да е “по-голям” от “елементът на offset2”.

Програмата трябва да интерпретира по ред дефинираните правила в comparator.bin и ако правилото не е изпълнено, да разменя in-place елементите на съответните отмествания. Елементи, които са равни, няма нужда да се разменят.

Зад. 36

Вашите колеги от съседната лаборатория са попаднали на вогонски крипто-вирус и вашата задача е да напишете програма, която възстановява файлове, криптирани от вируса.
Примерно извикване: ./main input.encrypted output.decrypted
Вирусът не бил особено добре проектиран и след анализ се оказало, че подменя оригиналните файлове с криптирани със следните особености:

- вогоните явно предпочитат да работят с данни от по 128 бита, като всеки такъв елемент ще наричаме unit (U). Както входния (оригинален) файл, така и изходния (криптиран) файл се третират като състоящи се от unit-и. Криптираният файл винаги е точен брой unit-и (номерирани спрямо N_0), като при нужда към оригиналният файл преди криптиране се добавят байтове 0x00 (padding), за да стане точен брой unit-и
- unit-ите в криптирания файл могат да се ползват за съхранение на следните типове данни (и техните големини): header (4U), section (2U), cryptdata (1U), unused data (1U)
- типовете данни section и cryptdata са обработени с побитово изключващо или (XOR) със 128 битов ключ (ключът е голям 1U), като за section ключът се ползва два пъти – за първият и за вторият unit по отделно
- криптираният файл винаги започва с header (4U) и има следната структура:
    - uint64_t, magic – магическа стойност 0x0000534f44614c47, която дефинира, че файлът е криптиран спрямо тази спецификация
    - uint32_t, cfsb – размер на криптирания файл в байтове
    - uint32_t, cfsu – размер на криптирания файл в брой unit-и
    - uint32_t, ofsb – размер на оригиналния файл в байтове
    - uint32_t, ofsu – размер на оригиналния файл в брой unit-и
    - uint32_t, unused1 – не се използва в тази версия на вируса
    - uint32_t, cksum – checksum-а на оригиналния файл, получена по следния алгоритъм:
        - оригиналният файл се третира като състоящ се от uint32_t елементи
        - при нужда от padding на последния елемент се ползват байтове 0x00
        - всичкиелементисеXOR-ватединсдругиполученията резултатът е cksum
    - 128 бита, sectionkey – ключ, с който са кодирани section-ите
    - четири “слота”, описващи начална позиция в U на section
        - uint32_t, s0
        - uint32_t, s1
        - uint32_t, s2
        - uint32_t, s3
        - тъй като unit-и с номера в интервала [0,3] се ползват за самия header, в горните слотове 0 означава, че този слот не се използва (няма такъв section)
        - unit-ите в изходния (декриптиран) файл са по реда на секциите
- section (2U) е кодиран със sectionkey от header-а и след разкодиране има следната структура:
    - int64_t, relative_offset – дефинира от кое U започват unit-ите с данни за тази секция. Числото е относително спрямо първият unit на section, например, ако section е записан в U 4/5 и стойността на offset е 2, то първите данни на секцията ще са в U 6
    - uint64_t, len – дефинира брой unit-и в този section
    - 128 бита, datakey – ключ, с който са кодирани unit-ите с данни в този section
    - unit-ите в изходния (декриптиран) файл са по реда на unit-ите в секцията
- cryptdata (1U) – unit в който има записани данни, кодирани с datakey на секцията, към която е този unit
- unused data (1U) – unit, който не се ползва. Колегите ви предполагат, че това е един от начините чрез които вогоните са вкарвали “шум” в криптирания файл, но без да правят декриптирането невъзможно.

Зад. 37
Напишете програма на C, която по подадено име на (текстови) файл като параметър, извежда съдържанието на файла сортирано, чрез употреба на външните програми cat и sort през pipe().

Зад. 38
Напишете програма на C, която реализира simple command prompt. Тя изпълнява в цикъл следната поредица действия:

- Извежда промпт на стандартния изход
- Прочита име на команда
- Изпълнява без параметри прочетената команда.

Командите се търсят в директорията /bin. За край на програмата се смята въвеждането на exit.

Зад. 39
Напишете програма на C, която използвайки външни shell команди през pipe() да извежда статистика за броя на използване на различните shell-ове от потребителите, дефинирани в системата. Изходът да бъде сортиран във възходящ ред според брой използвания на shell-овете.
Примерно извикване и изход:

```
$ ./main
        1 /bin/sync
        3 /bin/bash
        7 /bin/false
        17 /usr/sbin/nologin
```

Зад. 40
Напишете програма на C, която приема незадължителен параметър – име на команда. Ако не е зададена команда като параметър, да се ползва командата echo. Максималната допустима дължина на командата е 4 знака.
Програмата чете низове (с максимална дължина 4 знака) от стандартния си вход, разделени с интервали (0x20) или знак за нов ред (0x0A). Ако някой низ е с дължина по-голяма от 4 знака, то програмата да терминира със съобщение за грешка.
Подадените на стандартния вход низове програмата трябва да третира като множество от параметри за дефинираната команда. Програмата ви трябва да изпълни командата колкото пъти е необходимо с максимум два низа като параметри, като изчаква изпълнението да приключи, преди да започне ново изпълнение.
Примерни вход, извиквания и изходи:

```
$ cat f1
        a1
$ cat f2
        a2
$ cat f3
        a3
$ echo -e "f1\nf2 f3" | ./main cat
        a1
        a2
        a3
$ echo -e "f1\nf2 f3" | ./main
        f1
        f2 f3
```

Зад. 41
Напишете програма на C, която приема параметър – име на директория. Програмата трябва да извежда името на най-скоро променения (по съдържание) файл в тази директория и нейните под-директории, чрез употреба на външни шел команди през pipe().

Зад. 42
Напишете програма-наблюдател P, която изпълнява друга програма Q и я рестартира, когато Q завърши изпълнението си. На командния ред на P се подават следните параметри:

- праг за продължителност в секунди – едноцифрено число от 1 до 9
- Q
- незадължителни параметри на Q.

P работи по следния алгоритъм:

- стартира Q с подадените параметри
- изчаква я да завърши изпълнението си
- записва в текстов файл run.log един ред с три полета - цели числа (разделени с интервал):
    - момент на стартиране на Q (Unix time)
    - момент на завършване на Q (Unix time)
    - за грешка, с който Q е завършила (exit code)
- проверява дали е изпълнено условието за спиране и ако не е, преминава отново към стартирането на Q.

Условие за спиране: Ако наблюдателят P установи, че при две последователни изпълнения на Q са били изпълнени и двете условия:

- кодът за грешка на Q е бил различен от 0
- разликата между момента на завършване и момента на стартиране на Q е била по-малка от подадения като първи параметър на P праг,

то P спира цикъла от изпълняване на Q и сам завършва изпълнението си.
Текущото време във формат Unix time (секунди от 1 януари 1970 г.) можете да вземете с извикване на системната функция time() с параметър NULL; функцията е дефинирана в time.h. Ако изпълнената програма е била прекъсната от подаден сигнал, това се приема за завършване с код за грешка 129.

Зад. 43
Напишете две програми на C (foo и bar), които си комуникират през наименована тръба. Програмата foo приема параметър - име на файл, програмата bar приема параметър - команда като абсолютен път до изпълним файл.
Примерни извиквания и ред на изпълнение (в отделни терминали):

```
./foo a.txt
./bar /usr/bin/sort
```

Програмата foo трябва да изпълнява външна команда cat с аргумент името на подадения файл, така че съдържанието му да се прехвърли през тръбата към програмата bar, която от своя страна трябва да изпълни подадената и като аргумент команда (без параметри; /usr/bin/sort в примера), която да обработи получените през тръбата данни, четейки от стандартен вход. Еквивалент на горния пример би било следното изпълнение:

```
cat a.txt | /usr/bin/sort
```

Зад. 44

При изграждане на система за пренасяне на сериен асинхронен сигнал върху радиопреносна мрежа се оказало, че големи поредици от битове само нули или само единици смущават сигнала, поради нестабилно ниво. Инженерите решили проблема, като:

- в моментите, в които няма сигнал от серийният порт, вкарвали изкуствено байт 0x55 в потока
- реалните байтове 0x00, 0xFF, 0x55 и 0x7D се кодирали посредством XOR-ване (побитова обработка с изключващо-или) с 0x20, като полученият байт се изпращал през потока, предхождан от 0x7D, който играе ролята на escape character.

Разполагате със запис от такъв поток. Напишете програма на C, която приема два параметъра - имена на файлове.
Примерно извикване: $ ./main input.lfld output.bin
Програмата трябва да обработва записа и да генерира output.bin, който да съдържа оригиналните данни. Четенето на входните данни трябва да става посредством изпълнение на външна shell команда.

Зад. 45
Напишете програма на C, която приема един задължителен позиционен параметър - име на файл. Файлът се състои от не повече от 8 наредени тройки елементи:

- име на файл – точно 8 байта, последният от които задължително е 0x00. Ако името е по-късо от 7 знака, излишните байтове са 0x00
- offset – uint32_t, който дава пореден номер на елемент (спрямо N_0) във файла
- length – uint32_t, който дава брой елементи.

За всяка наредена тройка програмата трябва да пусне child процес, който да XOR-не (обработи с изключващо-или) елементите (uint16_t) от съответния файл един със друг, и да върне резултата на parent процеса, който от своя страна трябва да XOR-не всички получените резултати и да изведе полученото число в следния формат: result: 573B

Зад. 46
Ваши колеги - асистенти по ОС се оплакват, че студентите упорито ползват командата sudo, въпреки че им е обяснено да не я ползват. Вашата задача е да подготвите фалшиво sudo, което:

- добавя ред в текстови файл foo.log с командата, която потребителят се е опитал да изпълни във формат ДАТА ПОТРЕБИТЕЛ ДАННИ, където:
  - ДАТА е във формат YYYY-MM-DD HH:MM:SS.UUUUUU (UUUUUU са микросекунди)
  - ПОТРЕБИТЕЛ е login name (username) на потребителя
  - ДАННИ е всичко, което потребителят е написал, например ./main ls -l /root
  - горните полета са разделени с интервал
  - пример: 2021-05-10 14:38:17.584344 s99999 ./main ls -l /root
- заключва акаунта, който е изпълнил програмата
- приключва изпълнението на всички процеси, стартирани от този потребител.

За да може да изпълни тези неща, фалшивото sudo ще има нужда от root права, а в същото време трябва да знае кой акаунт изпълнява командата. Напишете програма на C (main), която за заключването на акаунта и терминирането на процесите извиква външни shell команди през pipe().
Опишете в как бихте настроили компилираната вече програма така, че студентите да ползват нея, вместо оригиналното sudo.

Зад. 47
Ваши колеги - асистенти по ОС имат нужда от демонстрационна програма на C, която да служи като пример за конкурентност и синхронизация на процеси. Напишете такава програма, приемаща два задължителни позиционни параметъра – едноцифрени числа.
Примерно извикване: ./main N D
Общ алгоритъм на програмата:

- началният (родителски) процес създава процес-наследник
- N на брой пъти се изпълнява:
  - родителският процес извежда на stdout низа “DING ”
  - процесът-наследник извежда на stdout низа “DONG”
  - родителският процес изчаква D секунди.

Гарантирайте, че:

- процесът-наследник винаги извежда “DONG” след като родителския процес е извел “DING ”
- родителският процес винаги започва изчакването след като процеса-наследник е извел “DONG”.

За изчакването погледнете sleep(3).

Зад. 48
Ваши колеги – асистенти по ОС – имат нужда от демонстрационна програма на C, която да служи като пример за конкурентност и синхронизация на процеси.
Дефиниран е нареден списък с три думи L = (”tic ” , ”tac ” , ”toe\n”), като всяка дума е с дължина четири знака. Напишете програма, приемаща две числа като аргументи (./main NC WC), като NC е ограничено в интервала [1,7], а WC – в интервала [1,35]. Програмата трябва задължително да работи по следния общ алгоритъм:

- началният (родителски) процес създава NC на брой процеси-наследници
- групата процеси извеждат на stdout общо WC на брой думи от горния списък при следните правила:

- ако броят на думите за извеждане WC е по-голям от общия брой думи в L, след изчерпване на думите в L програмата започва списъка отначало колкото пъти е нужно
- всеки процес извежда само една дума
- първата дума се извежда от родителския процес
- ако броят на думите за извеждане WC е по-голям от общия брой процеси, след изчерпване на процесите програмата започва да ги ползва от начало. Например при родителски процес P и два процеса-наследници C_1 и C_2 редът на използване на процесите ще бъде P, C_1, C_2, P, C_1, C_2, P, C_1, ...
- изведените думи гарантирано трябва да са по реда, дефиниран в L
- всеки процес гарантирано започва да извежда на stdout поредната дума, след като предходния процес е приключил с извеждането на предишната.