Примерни задачи за курс “Операционни системи”, СУ, ФМИ

14 април 2020 г.

Семинарните упражнения на курса “Операционни системи” разглеждат следните теми, групирани по “Работа в GNU/Linux shell” и “Използване на системни примитиви в програми на C”:

1. Въведение (shell)
2. Файлова система и работа с файлове (shell)
3. Обработка на текст (shell)
4. Процеси (shell)
5. Командни интерпретатори и скриптове (shell)
6. Системни примитиви за вход и изход (C)
7. Системни примитиви за работа с процеси (C)
8. Системни примитиви за работа с pipe-ове (C)

Тук са събрани примерни задачи по различните теми, както и някои теоретични въпроси, с надеждата те да бъдат полезни на студентите при тяхната работа в курса. Очаква се студентите да са разгледали някои по-базови задачи по дадена тема преди да преминат към изложените тук.

Работа в GNU/Linux shell

Забележка: За всички задачи, освен ако не е указано друго, в имената на файловете и директориите няма специални символи. Във файловата система може да съществуват директории, до които нямате достъп.

Задачи за теми 1,2,3

Зад. 1 Даден е текстов файл с име `philip-j-fry.txt`. Напишете shell script и/или серия от команди, които извеждат броя редове, съдържащи поне една четна цифра и несъдържащи малка латинска буква от `a` до `w`.

Примерно съдържание на файла:

```
123abv123
123zz123
MMU_2.4
```

Примерен изход:

```
Броят на търсените редовете е 2
```

Зад. 2 Имате текстов файл със следното съдържание (всяка книга е на един ред):

```
1979 г. - „Синият тайфун“ (сборник съветски научнофантастични разкази за морето)
1979 г. - „Двойната звезда“ - Любен Дилов
1979 г. - „Завръщане от звездите“ - Станислав Лем (Превод: Веселин Маринов)
1979 г. - „Среща с Рама“ - Артър Кларк (Превод: Александър Бояджиев)
1979 г. - „Алиби“ - Димитър Пеев (криминален роман)
1979 г. - „Тайнственият триъгълник“ (сборник НФ разкази за морето)
1979 г. - „Второто нашествие на марсианците“ - Аркадий и Борис Стругацки
```

```
1979 г. - „Гробищен свят“ - Клифърд Саймък (Превод: Михаил Грънчаров)
1979 г. - „Чоки“ - Джон Уиндъм (Превод: Теодора Давидова)
1979 г. - „Спускане в Маелстрьом“ - Едгар Алан По (Превод: Александър Бояджиев)
1980 г. - „Допълнителна примамка“ - Робърт Ф. Йънг (Превод: Искра Иванова, ...)
1980 г. - „Кристалното яйце“ - Хърбърт Уелс (Превод: Борис Миндов, ...)
1980 г. - „Онирофилм“ (сборник италиански НФ разкази) (Превод: Никола Иванов, ...)
```

Напишете shell script (приемащ аргумент име на файл) и серия от команди, които извеждат:

- всеки ред от файла с добавен пореден номер във формат "1. ", "2. ", ... "11. " ...
- махат данните за годината на издаване
- сортират изхода по заглавие (лексикографски, възходящо)

Примерен изход (показани са само първите 4 реда):

```
5. „Алиби“ - Димитър Пеев (криминален роман)
7. „Второто нашествие на марсианците“ - Аркадий и Борис Стругацки
8. „Гробищен свят“ - Клифърд Саймък (Превод: Михаил Грънчаров)
2. „Двойната звезда“ - Любен Дилов
```

Зад. 3 В текущата директория има само обикновени файлове (без директории). Да се напише bash script, който приема 2 позиционни параметъра – числа, който мести файловете от текущата директория към нови директории (a, b и c, които трябва да бъдат създадени), като определен файл се мести към директория 'a', само ако той има по-малко редове от първи позиционен параметър, мести към директория 'b', ако редове са между първи и втори позиционен параметър и в 'c' в останалите случаи.

Зад. 4 Файловете във вашата home директория съдържат информация за музикални албуми и имат специфична структура. Началото на всеки ред е годината на издаване на албума, а непосредствено, след началото на всеки ред следва името на изпълнителя на песента. Имената на файловете се състоят от една дума, която съвпада с името на изпълнителя.

Примерно съдържание на файл с име "Bonnie":

```
2005г. Bonnie - "God Was in the Water" (Randall Bramblett, Davis Causey) - 5:17
2005г. Bonnie - "Love on One Condition" (Jon Cleary) - 3:43
2005г. Bonnie - "So Close" (Tony Arata, George Marinelli, Pete Wasner) - 3:22
2005г. Bonnie - "Trinkets" (Emory Joseph) - 5:02
2005г. Bonnie - "Crooked Crown" (David Batteau, Maia Sharp) - 3:49
2005г. Bonnie - "Unnecessarily Mercenary" (Jon Cleary) - 3:51
2005г. Bonnie - "I Will Not Be Broken" - "Deep Water" (John Capek, Marc Jordan) - 3:58
```

Да се състави процедура на bash приемаща два параметъра, които са имена на файлове от вашата home директория. Скриптът сравнява, кой от двата файла има повече на брой редове, съдържащи неговото име (на файла). За файлът победител изпълнете следните действия:

- извлечете съдържанието му, без годината на издаване на албума и без името на изпълнителя
- сортирайте лексикографски извлеченото съдържание и го запишете във файл с име 'изпълнител.songs'

Примерен изходен файл (с име Bonnie.songs):

```
"Crooked Crown" (David Batteau, Maia Sharp) - 3:49
"God Was in the Water" (Randall Bramblett, Davis Causey) - 5:17
"I Will Not Be Broken" - "Deep Water" (John Capek, Marc Jordan) - 3:58
"Love on One Condition" (Jon Cleary) - 3:43
"So Close" (Tony Arata, George Marinelli, Pete Wasner) - 3:22
"Trinkets" (Emory Joseph) - 5:02
"Unnecessarily Mercenary" (Jon Cleary) - 3:51
```

Зад. 5 Напишете серия от команди, извеждащи на екрана само броя на всички обекти във файловата система, чиито собственик е текущият потребител.

Забележка: Във файловата система със сигурност съществуват директории, до които нямате достъп.

Зад. 6 Напишете серия от команди, които изтриват:

а) всички файлове в текущата директория и нейните поддиректории, които са с нулева дължина.
б) 5-е най-големи файла в home директорията на текущия потребител и нейните поддиректории.

Зад. 7 Напишете серия от команди, които от файла `/etc/passwd` да вземат под-низ, състоящ се от втора и трета цифра на факултетния номер на студентите от специалност Информатика, чиито фамилии завършват на "а". Изведете коя комбинация от цифри се среща най-често и коя е тя.

Примерно съдържание на файла:

```
s45194:x:1255:502:Elizabet Mihaylova, Inf, k3, g1:/home/Inf/s45194:/bin/bash
s45139:x:1261:502:Vasilena Peycheva:/home/Inf/s45139:/bin/bash
s81257:x:1079:503:Vasilena Nikolova, KN, 2kurs, 5gr:/home/KN/s81257:/bin/bash
s81374:x:1117:503:Ivan Kamburov, KN, 2kurs, 7gr:/home/KN/s81374:/bin/bash
kiril:x:508:500:Kiril Varadinov:/home/kiril:/bin/bash
s61812:x:1128:504:Vladimir Genchev:/home/SI/s61812:/bin/bash
user:x:1000:99:Inactive user just to start UID from 1000:/home/user:/sbin/nologin
s81254:x:1077:503:Mariela Tihova, KN, 2kurs, 5gr:/home/KN/s81254:/bin/bash
s81386:x:1121:503:Daniela Ruseva, KN, 2kurs, 7gr:/home/KN/s81386:/bin/bash
s45216:x:1235:502:Aleksandar Yavashev, Inf, k3, g3:/home/Inf/s45216:/bin/bash
```

Примерен изход:

```
2 51
```

Зад. 8 Намерете имената на топ 5 файловете в текущата директория с най-много hardlinks.

Зад. 9 Напишете серия от команди, извеждащи на екрана *само* inode-а на най-скоро променения (по съдържание) файл, намиращ се в home директорията на потребител pesho (или нейните под-директории), който има повече от едно име.

Зад. 10 При подреждане в нарастващ ред на числовите потребителски идентификатори (`UID`) на акаунтите, дефинирани в системата, 201-ят акаунт е от групата, запазена за акаунти от специалност СИ.

Изведете списък с имената (име и фамилия) и home директориите на всички акаунти от специалност СИ, подреден по факултетен номер.

За справка:

```
s61988:x:1219:504:Stoian Genchev,SI,2,5:/home/SI/s61988:/bin/bash
s81430:x:1234:503:Iordan Petkov, KN, k2, g7:/home/KN/s81430:/bin/bash
s61807:x:1248:504:Elica Venchova:/home/SI/s61807:/bin/bash
s62009:x:1254:504:Denitsa Dobreva, 2, 6:/home/SI/s62009:/bin/bash
s61756:x:1258:504:Katrin Kartuleva, SI, 4, 1:/home/SI/s61756:/bin/bash
s855287:x:1195:504:Vaska Kichukova,SI,2,5:/home/SI/s855287:/bin/bash
```

Примерен изход:

```
Katrin Kartuleva:/home/SI/s61756
Elica Venchova:/home/SI/s61807
Stoian Genchev:/home/SI/s61988
Denitsa Dobreva:/home/SI/s62009
Vaska Kichukova:/home/SI/s855287
```

Зад. 11 Вие сте асистент по ОС. На първото упражнение казвате на студентите да си напишат данните на лист, взимате го и им правите акаунти. След упражнението обаче, забравяте да вземете листа със себе си - сещате се половин час по-късно, когато трябва да въведете имената на студентите в

таблица, но за зла беда в стаята вече няма ни помен от листа (вероятно иззет от спешния отряд на GDPR-полицията)

Сещате се, че в началото на упражнението UNIX-часовникът е показвал 1551168000, а в края 1551176100.

Напишете команда, която изкарва разделени с таб факултетните номера и имената на потребителите от специалност СИ, чиито home директории са променили статуса си (status change time) в зададения времеви интервал.

Приемете, че всички потребители от СИ имат home директории под /home/SI.

Примерен изход:

```
62198   Ivaylo Georgiev
62126   Victoria Georgieva
62009   Denitsa Dobreva
62208   Trayana Nedelcheva
```

Няколко реда от `/etc/passwd` за справка:

```
s62136:x:1302:503:Alexander Ignatov, SI, 2, 2:/home/KN/s62136:/bin/bash
s62171:x:1031:504:Deivid Metanov:/home/SI/s62171:/bin/bash
s62126:x:1016:504:Victoria Georgieva:/home/SI/s62126:/bin/bash
s62009:x:1170:504:Denitsa Dobreva,SI,3,3:/home/SI/s62009:/bin/bash
s62196:x:1221:504:Elena Tuparova,SI,2,1:/home/SI/s62196:/bin/bash
```

Зад. 12 От всички файлове в home директорията на потребителя velin, изведете дълбочината на файл, който:

- има същия inode като този на най-скоро променения файл сред тях
- има минимална дълбочина

Пояснение Под "дълбочина"да се разбира дълбочина в дървото на файловата система: например файлът `/foo/bar/baz` има дълбочина 3.

Задачи за теми 1,2,3,4,5

Зад. 13 Напишете shell скрипт, който по подаден един позиционен параметър, ако този параметър е директория, намира всички symlink-ове в нея и под-директориите ѝ с несъществуващ destination.

Зад. 14 Напишете shell скрипт, който приема един позиционен параметър - число. Ако скриптът се изпълнява като root, да извежда обобщена информация за общото количество активна памет *(RSS - resident set size, non-swapped physical memory that a task has used)* на процесите на всеки потребител. Ако за някой потребител обобщеното число надвишава подадения параметър, да изпраща подходящи сигнали за прекратяване на процеса с най-много активна памет на потребителя.

Забележка: Приемаме, че изхода в колоната RSS е число в същата мерна единица, като числото, подадено като аргумент. Примерен формат:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  15816  1884 ?        Ss   May12   0:03 init [2]
root         2  0.0  0.0      0     0 ?        S    May12   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    May12   0:02 [ksoftirqd/0]
```

Забележка: Алтернативно може да ползвате изхода от `ps -e -o uid,pid,rss`

Зад. 15 Напишете shell скрипт който, ако се изпълнява от root, проверява кои потребители на системата нямат homedir или не могат да пишат в него.

Примерен формат:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

Зад. 16 Напишете скрипт, който приема три задължителни позиционни аргумента:

- име на фаил
- *низ1*
- *низ2*

Файлът е текстови, и съдържа редове във формат:

ключ=*стойност*

където *стойност* може да бъде:

- празен низ, т.е. редът е *ключ*=
- низ, състоящ се от един или повече термове, разделени с интервали, т.е., редът е *ключ*=t_1 t_2 t_3

Някъде във файла:

- се съдържа един ред с *ключ* първия подаден низ (*низ1*);
- и може да се съдържа един ред с *ключ* втория подаден низ (*низ2*).

Скриптът трябва да променя реда с *ключ низ2* така, че обединението на термовете на редовете с ключове *низ1* и *низ2* да включва всеки терм еднократно.

Примерен входен файл:

```
$ cat z1.txt
FOO=73
BAR=42
BAZ=
ENABLED_OPTIONS=a b c d
ENABLED_OPTIONS_EXTRA=c e f
```

Примерно извикване:

```
$ ./a.sh z1.txt ENABLED_OPTIONS ENABLED_OPTIONS_EXTRA
```

Изходен файл:

```
$ cat z1.txt
FOO=73
BAR=42
BAZ=
ENABLED_OPTIONS=a b c d
ENABLED_OPTIONS_EXTRA=e f
```

Зад. 17 Напишете скрипт, който приема задължителен позиционен аргумент - име на потребител *FOO*. Ако скриптът се изпълнява от root:

а) да извежда имената на потребителите, които имат повече на брой процеси от *FOO*, ако има такива;
б) да извежда средното време (в секунди), за което са работили процесите на всички потребители на системата (`TIME`, във формат *HH:MM:SS*);
в) ако съществуват процеси на *FOO*, които са работили над два пъти повече от средното време, скриптът да прекратява изпълнението им по подходящ начин.

За справка:

```
$ ps -e -o user,pid,%cpu,%mem,vsz,rss,tty,stat,time,command | head -5
USER       PID %CPU %MEM    VSZ   RSS TT       STAT     TIME COMMAND
root         1  0.0  0.0  15820  1920 ?        Ss   00:00:05 init [2]
root         2  0.0  0.0      0     0 ?        S    00:00:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    00:00:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   00:00:00 [kworker/0:0H]
```

Зад. 18 Напишете скрипт, който извежда името на потребителския акаунт, в чиято home директория има най-скоро променен обикновен файл и кой е този файл. Напишете скрипта с подходящите проверки, така че да бъде валиден инструмент.

Зад. 19 Напишете скрипт, който получава задължителен първи позиционен параметър – директория и незадължителен втори – число. Скриптът трябва да проверява подадената директория и нейните под-директории и да извежда имената на:

а) при подаден на скрипта втори параметър – всички файлове с брой hardlink-ове поне равен на параметъра;
б) при липса на втори параметър – всички symlink-ове с несъществуващ destination (счупени symlink-ове).

*Забележка:*За удобство приемаме, че ако има подаден втори параметър, то той е число.

Зад. 20 Напишете скрипт, който приема три задължителни позиционни параметра - директория *SRC*, директория *DST* (която не трябва да съдържа файлове) и низ *ABC*. Ако скриптът се изпълнява от root потребителя, то той трябва да намира всички файлове в директорията *SRC* и нейните под-директории, които имат в името си като под-низ *ABC*, и да ги мести в директорията *DST*, запазвайки директорийната структура (но без да запазва мета-данни като собственик и права, т.е. не ни интересуват тези параметри на новите директории, които скриптът би генерирал в *DST*).

Пример:

- в *SRC* (`/src`) има следните файлове:

  ```
  /src/foof.txt
  /src/1/bar.txt
  /src/1/foo.txt
  /src/2/1/foobar.txt
  /src/2/3/barf.txt
  ```

- *DST* (`/dst`) е празна директория
- зададения низ е `foo`

Резултат:

- в *SRC* има следните файлове:

  ```
  /src/1/bar.txt
  /src/2/3/barf.txt
  ```

- в *DST* има следните файлове:

  ```
  /dst/foof.txt
  /dst/1/foo.txt
  /dst/2/1/foobar.txt
  ```

Зад. 21 Напишете скрипт, който ако се изпълнява от root потребителя:

а) извежда обобщена информация за броя и общото количество активна памет *(RSS - resident set size, non-swaped physical memory that a task has used)* на текущите процеси на всеки потребител;
б) ако процесът с най-голяма активна памет на даден потребител използва два пъти повече памет от средното за потребителя, то скриптът да прекратява изпълнението му по подходящ начин.

За справка:

```
$ ps aux | head -3
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  15820  1052 ?        Ss   Apr21   0:06 init [2]
root         2  0.0  0.0      0     0 ?        S    Apr21   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Apr21   0:02 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Apr21   0:00 [kworker/0:0H]
```

Алтернативно, може да ползвате изхода от `ps -e -o uid,pid,rss`

Зад. 22 Напишете shell script, който получава задължителен първи позиционен параметър – директория и незадължителен втори – име на файл. Скриптът трябва да намира в подадената директория и

нейните под-директории всички symlink-ове и да извежда (при подаден аргумент файл – добавяйки към файла, а ако не е – на стандартния изход) за тях следната информация:

- ако destination-а съществува – *името на symlink-а* `->` *името на destination-а*;
- броя на symlink-овете, чийто destination не съществува.

Примерен изход:

```
lbaz -> /foo/bar/baz
lqux -> ../../../qux
lquux -> /foo/quux
Broken symlinks: 34
```

Зад. 23 Напишете скрипт, който получава два задължителни позиционни параметъра – директория и низ. Сред файловете в директорията би могло да има такива, чиито имена имат структура `vmlinuz-`*x*`.`*y*`.`*z*`-`*arch* където:

- `vmlinuz` е константен низ;
- тиретата "`-`" и точките "`.`" присъстват задължително;
- *x* е число, version;
- *y* е число, major revision;
- *z* е число, minor revision;
- наредената тройка *x.y.z* формира глобалната версия на ядрото;
- *arch* е низ, архитектура (платформа) за която е съответното ядро.

Скриптът трябва да извежда само името на файла, намиращ се в подадената директория (но не и нейните поддиректории), който:

- спазва гореописаната структура;
- е от съответната архитектура спрямо параметъра-низ, подаден на скрипта;
- има най-голяма глобална версия.

Пример:

- Съдържание на `./kern/`:

  ```
  vmlinuz-3.4.113-amd64
  vmlinuz-4.11.12-amd64
  vmlinuz-4.12.4-amd64
  vmlinuz-4.19.1-i386
  ```

- Извикване и изход:

  ```
  $ ./task1.sh ./kern/ amd64
  vmlinuz-4.12.4-amd64
  ```

Зад. 24 Напишете скрипт, който ако се изпълнява от root потребителя, намира процесите на потребителите, които не са root потребителя и е изпълнено поне едно от следните неща:

- имат зададена несъществуваща home директория;
- не са собственици на home директорията си;
- собственика на директорията не може да пише в нея.

Ако общото количество активна памет *(RSS - resident set size, non-swaped physical memory that a task has used)* на процесите на даден такъв потребител е по-голямо от общото количество активна памет на root потребителя, то скриптът да прекратява изпълнението на всички процеси на потребителя.

За справка:

```
$ ps aux | head -3
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  15820  1052 ?        Ss   Apr21   0:06 init [2]
root         2  0.0  0.0      0     0 ?        S    Apr21   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Apr21   0:02 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Apr21   0:00 [kworker/0:0H]
```

Алтернативно, може да ползвате изхода от `ps -e -o uid,pid,rss`

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
s61934:x:1177:504:Mariq Cholakova:/home/SI/s61934:/bin/bash
```

Зад. 25 Нека съществува програма за моментна комуникация (Instant messaging), която записва логове на разговорите в следния формат:

- има определена директория за логове (*LOGDIR*)
- в нея има директорийна структура от следния вид:
  LOGDIR/протокол/акаунт/приятел/
  като на всяко ниво може да има няколко екземпляра от съответния вид, т.е. няколко директории *протокол*, във всяка от тях може да има няколко директории *акаунт*, и във всяка от тях – няколко директории *приятел*
- във всяка от директориите *приятел* може да има файлове с имена от вида `yyyy-mm-dd-hh-mm-ss.txt` – година-месец-ден и т.н., спрямо това кога е започнал даден разговор
- всеки такъв файл представлява лог на даден разговор със съответния приятел, като всяка разменена реплика между вас е на отделен ред
- даден идентификатор *приятел* може да се среща няколко пъти в структурата (напр. през различни ваши акаунти сте водили разговори със същия приятел)

Напишете скрипт, който приема задължителен позиционен аргумент - име на лог директория (*LOGDIR*). Скриптът трябва да извежда десетимата приятели, с които имате най-много редове комуникация глобално (без значение протокол и акаунт), и колко реда имате с всеки от тях. Опишете в коментар как работи алгоритъмът ви.

Зад. 26 Напишете скрипт, който приема два позиционни аргумента – име на текстови файл и директория. Директорията не трябва да съдържа обекти, а текстовият файл (US-ASCII) е стенограма и всеки ред е в следния формат:

```
ИМЕ ФАМИЛИЯ (уточнения): Реплика
```

където:

- `ИМЕ ФАМИЛИЯ` присъстват задължително;
- `ИМЕ` и `ФАМИЛИЯ` се състоят само от малки/главни латински букви и тирета;
- `(уточнения)` не е задължително да присъстват;
- двоеточието ‘`:`’ присъства задължително;
- `Репликата` не съдържа знаци за нов ред;
- в стринга преди двоеточието ‘`:`’ задължително има поне един интервал между `ИМЕ` и `ФАМИЛИЯ`;
- наличието на други интервали където и да е на реда е недефинирано.

Примерен входен файл:

```
John Lennon (The Beatles): Time you enjoy wasting, was not wasted.
Roger Waters: I'm in competition with myself and I'm losing.
John Lennon:Reality leaves a lot to the imagination.
Leonard  Cohen:There is a crack in everything, that's how the light gets in.
```

Скриптът трябва да:

- създава текстови файл `dict.txt` в посочената директория, който на всеки ред да съдържа:

  ```
  ИМЕ ФАМИЛИЯ;НОМЕР
  ```

  където:
  - `ИМЕ ФАМИЛИЯ` е уникален участник в стенограмата (без да се отчитат уточненията);
  - `НОМЕР` е уникален номер на този участник, избран от вас.
- създава файл `НОМЕР.txt` в посочената директория, който съдържа всички (и само) редовете на дадения участник.

Зад. 27 Напишете скрипт, който приема два позиционни аргумента – имена на текстови файлове в `CSV` формат:

```
8,foo,bar,baz
```

```
2,quz,,foo
12,1,3,foo
3,foo,,
5,,bar,
7,,,
4,foo,bar,baz
```

Валидни са следните условия:

- CSV файловете представляват таблица, като всеки ред на таблицата е записан на отделен ред;
- на даден ред всяко поле (колона) е разделено от останалите със запетая;
- броят на полетата на всеки ред е константа;
- в полетата не може да присъства запетая, т.е., запетаята винаги е разделител между полета;
- ако във файла присъстват интервали, то това са данни от дадено поле;
- първото поле на всеки ред е число, което представлява идентификатор на реда (ID).

Примерно извикване: `./foo.sh a.csv b.csv`

Скриптът трябва да чете `a.csv` и на негова база да създава `b.csv` по следния начин:

- някои редове във файла се различават само по колоната ID, и за тях казваме, че формират множество A_i
- за всяко такова множество A_i да се оставя само един ред - този, с най-малка стойност на ID-то;
- редовете, които не са членове в някое множество A_i се записват в изходния файл без промяна.

Зад. 28 Напишете два скрипта (по един за всяка подточка), които четат редове от STDIN. Скриптовете трябва да обработват само редовете, които съдържат цели положителни или отрицателни числа; останалите редове се игнорират. Скриптовете трябва да извежда на STDOUT:

а) всички уникални числа, чиято абсолютна стойност е равна на максималната абсолютна стойност сред всички числа
б) всички най-малки уникални числа от тези, които имат максимална сума на цифрите си

Примерен вход:

```
We don't
n11d n0
educat10n
12.3
6
33
-42
-42
111
111
-111
```

Примерен изход за а):

```
-111
111
```

Примерен изход за б):

```
-42
```

Зад. 29 Напишете шел скрипт, който приема множество параметри. Общ вид на извикване:

```
./foo.sh [-n N] FILE1...
```

В общия случай параметрите се третират като имена на (.log) файлове, които трябва да бъдат обработени от скрипта, със следното изключение: ако първият параметър е стрингът -n, то вторият параметър е число, дефиниращо стойност на променливата N, която ще ползваме в скрипта. Въвеждаме понятието *идентификатор на файл* (ИДФ), което се състои от името на даден файл без разширението .log. За удобство приемаме, че скриптът:

- ще бъде извикван с аргументи имена на файлове, винаги завършващи на `.log`
- няма да бъде извикван с аргументи имена на файлове с еднакъв ИДФ.

Лог файловете са текстови, като всеки ред има следния формат:

- време: timestamp във формат `YYYY-MM-DD HH:MM:SS`
- интервал
- данни: поредица от символи с произволна дължина

За удобство приемаме, че редовете във всеки файл са сортирани по време възходящо.

Примерно съдържание на даден лог файл:

```
2019-05-05 06:26:54 orthanc rsyslogd:  rsyslogd was HUPed
2019-05-06 06:30:32 orthanc rsyslogd:  rsyslogd was HUPed
2019-05-06 10:48:29 orthanc kernel: [1725379.728871] Chrome_~dThread[876]: segfault
```

Скриптът трябва да извежда на `STDOUT` последните N реда (ако N не е дефинирано - 10 реда) от всеки файл, в следния формат:

- timestamp във формат `YYYY-MM-DD HH:MM:SS`
- интервал
- ИДФ
- интервал
- данни

Изходът трябва да бъде глобално сортиран по време възходящо.

Зад. 30 За удобство приемаме, че разполагате със системен инструмент sha256sum, който приема аргументи имена на файлове като за всеки файл пресмята и извежда уникална хеш стойност, базирана на съдържанието на файла. Изходът от инструмента е текстови, по един ред за всеки подаден като аргумент файл, в следния формат:

- хеш стойност с дължина точно 64 знака
- два интервала
- име на файл

Примерна употреба и изход:

```
$ sha256sum /var/log/syslog /var/log/user.log README.md
b2ff8bd882a501f71a144b7c678e3a6bc6764ac48eb1876fb5d11aac11014b78  /var/log/syslog
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  /var/log/user.log
e4702d8044b7020af5129fc69d77115fd4306715bd678ba4bef518b2edf01fb9  README.md
```

Напишете скрипт, който приема задължителен параметър име на директория (ДИР1). Някъде в директорията ДИР1 може да съществуват архивни файлове с имена `NAME_report-TIMESTAMP.tgz`, където:

- `NAME` е низ, който не съдържа символ `'_'`
- `TIMESTAMP` е във формат Unix time (POSIX time/UNIX Epoch time)

На всяко пускане на скрипта се обработват само новосъздадените или модифицираните по съдържание спрямо предното пускане на скрипта архивни файлове от горния тип. За всеки такъв архивен файл се изпълнява следното:

- ако архивният файл съдържа файл с име meow.txt, то този текстови файл да бъде записан под името `/extracted/NAME_TIMESTAMP.txt`, където `NAME` и `TIMESTAMP` са съответните стойности от името на архивния файл.

Теоретични задачи

Зад. 31 Всеки от процесите P и Q изпълнява поредица от три инструкции:

```
process P                process Q
  p_1                      q_1
```

```
p_2                    q_2
p_3                    q_3
```

Осигурете чрез семафори синхронизация на P и Q така, че инструкция p_1 да се изпълни преди q_2, а q_2 да се изпълни преди p_3.

Зад. 32 Опишете накратко основните процедури и структури данни, необходими за реализация на семафор.

Каква е разликата между слаб и силен семафор?

Опишете максимално несправедлива ситуация, която може да се получи в избирателна секция, ако на входа на секцията пазач – член на изборната комисия пуска гласоподавателите вътре така:

(1) във всеки момент в секцията може да има най-много двама гласоподаватели.

(2) пазачът работи като слаб семафор.

Зад. 33

Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P              process Q              process R
  p_1                    q_1                    r_1
  p_2                    q_2                    r_2
  p_3                    q_3                    r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че инструкция p_1 да се изпълни преди q_2 и r_2.

Забележка: Решения на задачата с повече от един семафор носят не повече от 20 точки.

Зад. 34 Преди стартиране на процеси P и Q са инициализирани два семафора и брояч:

```
semaphore e, m
e.init(1); m.init(1)
int cnt = 0
```

Паралелно работещи няколко копия на всеки от процесите P и Q изпълняват поредица от инструкции:

```
process P                    process Q
  m.wait()                     e.wait()
    cnt=cnt+1                    q_section
    if cnt=1 e.wait()          e.signal()
  m.signal()

  p_section

  m.wait()
    cnt=cnt-1
    if cnt=0 e.signal()
  m.signal()
```

Дайте обоснован отговор на следните въпроси:

а) Могат ли едновременно да се изпълняват инструкциите $p_{section}$ и $q_{section}$?
б) Могат ли едновременно да се изпълняват няколко инструкции $p_{section}$?
в) Могат ли едновременно да се изпълняват няколко инструкции $q_{section}$?
г) Има ли условия за deadlock или starvation за някой от процесите?

Упътване:

- Ще казваме, че P е в критична секция, когато изпълнява инструкцията си $p_{section}$. Същото за Q, когато изпълнява $q_{section}$.
- Изяснете смисъла на брояча *cnt* и какви процеси могат да бъдат приспани в опашките на двата семафора.

- Покажете, че в опашката на семафора e има най-много едно копие на P и произволен брой копия на Q.
- Покажете, че в момента на изпълнение на $e.signal()$ в кой да е от процесите, никой процес не е в критичната си секция.

Зад. 35

а) Няколко копия на процеса P изпълняват поредица от три инструкции:

```
process P
  p_1
  p_2
  p_3
```

Осигурете чрез семафор синхронизация на копията така, че най-много един процес да изпълнява инструкция p_2 във всеки един момент.

б) Опишете разликата при реализация на слаб и силен семафор.

в) Възможно ли е в зависимост от начина на реализация на семафора в подусловие а) да настъпят условия за deadlock или starvation? Ако да, опишете сценарий за поява на неприятната ситуация.

Зад. 36 Всеки от процесите P и Q изпълнява поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на P и Q, така че инструкция p_1 да се изпълни преди q_2, а q_1 да се изпълни преди p_2.

Зад. 37 Всеки от процесите P и Q изпълнява поредица от три инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
  p_3                      q_3
```

Осигурете чрез два семафора синхронизация на P и Q така, че отделните инструкции да се изпълнят в следния времеви ред: $p_1, q_1, p_2, q_2, p_3, q_3$

Зад. 38 Да приемем, че в съвременната операционна система процесът има 4 състояния:

- R – работещ (*running*, използва CPU)
- A – активен (*ready*, очаква CPU)
- S – блокиран (*sleeping*, очаква вход/изход)
- T – изчакващ време (*sleeping*, очаква времеви момент)

Нарисувайте диаграма на състоянията и преходите между тях. Диаграмата е ориентиран граф с върхове отделните състояния и ребра – възможните преходи.

Опишете накратко събитията, предизвикващи преход по всяко ребро на графа.

Зад. 39 Процесът P създава тръба (pipe) с извикване на функцията `pipe(int pipefd[2])` в ОС GNU/Linux.

а) Кои процеси не могат да ползват тръбата?

б) Опишете друг метод за изграждане на комуникационен канал, който дава възможност на произволни процеси да изградят и ползват канала. Допълнително искаме новоизградения канал да е достъпен само за процесите, които са го създали.

Упътване: Прочетете man-страницата за функцията `pipe()`.

Зад. 40 Множество паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на работещите копия, така че:

а) В произволен момент от времето да работи най-много едно от копията.
б) Работещите копия да се редуват във времето – след изпълнение на копие на P да следва изпълнение на копие на Q и обратно.
в) Първоначално е разрешено да се изпълни копие на P.

Зад. 41 Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят едновременно следните изисквания:

- Инструкция p_1 да се изпълни преди q_2 и r_2.
- Инструкция r_2 да се изпълни преди p_3.

Забележка: Решение с 2 семафора ще бъде оценено с 30 точки, решение с повече семафори ще ви донесе 20 точки.

Зад. 42 Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят едновременно следните изисквания:

- Инструкция p_1 да се изпълни преди q_2.
- Инструкция q_1 да се изпълни преди r_2.
- Инструкция r_1 да се изпълни преди p_2.
- Инструкция r_3 да се изпълни след p_2 и q_2.

Зад. 43 При споделено ползване на памет от няколко процеса е възможно да настъпи надпревара за ресурси (race condition).

(а – 10 точки) Дефинирайте понятието race condition.

(б – 5 точки) Възможно ли е да настъпи race condition в еднопроцесорна система? Ако да, при какви условия.

(в – 15 точки) Какви инструменти ползваме, за да избегнем race condition?

Зад. 44 Една от класическите задачи за синхронизация се нарича *Задача за читателите и писателите* (*Readers-writers problem*).

а) (10 точки) Опишете условието на задачата.
б) (20 точки) Опишете решение, използващо семафори.

Зад. 45 Какви са възможните състояния на процес.

Нарисувайте диаграма на състоянията и преходите между тях.

Опишете накратко ситуациите, предизвикващи преходи между състояния.

Зад. 46

Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят едновременно следните изисквания:

- Някоя от инструкциите p_2 и q_2 да се изпълни преди r_2.
- Ако инструкция p_2 се изпълни преди r_2, то q_2 да се изпълни след r_2.
- Ако инструкция q_2 се изпълни преди r_2, то p_2 да се изпълни след r_2.

Забележка: Решение с 2 семафора ще бъде оценено с 30 точки, решение с повече семафори ще ви донесе 20 точки.

Зад. 47 Дадена е програма за ОС Linux, написана на езика C:

```
#include <unistd.h>
#include <stdio.h>
int main(void)
{
  int p1, p2;
  p1=fork();
  p2=fork();
  printf("Hello world!\n");
}
```

а) Колко пъти ще се отпечата текста `"Hello world!"` при изпълнението на програмата? Обосновете отговора си.
б) Как работи системното извикване `fork()`?
в) Нарисувайте кореновото дърво с върхове процесите, които ще се стартират в резултат от изпълнението на програмата и ребра двойките родител-наследник.

Зад. 48 Множество паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от три инструкции:

```
process P          process Q
  p_1                q_1
  p_2                q_2
  p_3                q_3
```

Осигурете чрез семафори синхронизация на работещите копия, така че три инструкции – p_1, q_2 и p_3 се редуват циклично:

- първа се изпълнява инструкция p_1 на някое от работещите копия на процес P;
- след завършването ѝ се изпълнява инструкция q_2 на някое копие на Q;
- след нея – p_3 на някое копие на P;
- с това едно минаване през цикъла завършва и отново може да се изпълни инструкция p_1 на някое от работещите копия на процес P.

Зад. 49 Опишете накратко кои системни извиквания изграждат стандартните комуникационни канали в UNIX – неименувана тръба (pipe), връзка процес-файл, двустранна връзка процес-процес (connection).

Зад. 50 Опишете какви изисквания удовлетворява съвременна файлова система, реализирана върху блоково устройство (*block device*). Опишете накратко реализацията и целта на следните инструменти:

а) отлагане на записа, алгоритъм на асансьора;
б) поддържане на журнал на файловата система.

Зад. 51 При реализация на файлова система върху твърд диск файловете и директориите се записват върху сектори от диска. Времето за достъп до секторите зависи от текущото положение на механичните компоненти на диска – над коя пътечка е главата за четене/запис и каква е позицията ѝ над пътечката.

Защо се прави разместване във времето на операциите по четене и запис върху диска?

Опишете накратко реализацията и целта на алгоритъма на асансьора.

Зад. 52 Дадена е програма за ОС Linux, написана на езика C:

```
#include <unistd.h>
```

```
#include <stdio.h>
int main(void)
{
  int p1, p2, p3;
  p1=fork();
  if (p1==0) {
    p2=fork();
    if (p2>0) p3=fork();
  }
  printf("Hello world!\n");
}
```

а) Колко пъти ще се отпечата текста `Hello world!` при изпълнението на програмата? Обосновете отговора си.
б) Как работи системното извикване `fork()`?
в) Нарисувайте кореновото дърво с върхове процесите, които ще се стартират в резултат от изпълнението на програмата и ребра двойките родител-наследник.

Зад. 53 Опишете как се изгражда комуникационен канал (connection) между процес-сървер и процес-клиент със следните системни извиквания в стандарта POSIX:

socket(), bind(), connect(), listen(), accept()

Зад. 54 Опишете накратко основните комуникационни канали в ОС Linux.

Кои канали използват пространството на имената и кои не го правят?

Зад. 55 Всеки от процесите P и Q изпълнява поредица от три инструкции:

```
process P              process Q
  p_1                    q_1
  p_2                    q_2
  p_3                    q_3
```

Осигурете чрез два семафора синхронизация на P и Q така, че да са изпълнени едновременно следните времеви зависимости:

1. инструкция p_1 да се изпълни преди q_2
2. инструкция q_2 да се изпълни преди p_3
3. инструкция q_1 да се изпълни преди p_2
4. инструкция p_2 да се изпълни преди q_3

Забележка: За решение с повече семафори ще получите 20 точки.

Зад. 56 Множество паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от две инструкции:

```
process P              process Q
  p_1                    q_1
  p_2                    q_2
```

Осигурете чрез семафори синхронизация на работещите копия, така че да са изпълнени едновременно следните условия:

- В произволен момент от времето да работи най-много едно от копията.
- Работещите копия да се редуват във времето – след изпълнение на копие на P да следва изпълнение на копие на Q и обратно.
- Първоначално е разрешено да се изпълни копие на P.

Зад. 57 Процесите P и Q се изпълняват паралелно. Споделената променлива A има начална стойност 4. Променливата R е локална за двата процеса.

```
process P              process Q
  R=A                    R=A
  R=R+3                  R=R+2
  A=R                    A=R
```

Каква е стойността на A след изпълнението на процесите? Дайте обоснован отговор.

Зад. 58 Опишете разликата между синхронни и асинхронни входно-изходни операции. Дайте примери за програми, при които се налага използването на асинхронен вход-изход.

Зад. 59 Множество паралелно работещи копия на процеса P изпълняват поредица от две инструкции:

```
process P
  p_1
  p_2
```

Осигурете чрез семафори синхронизация на работещите копия, така че:

- Инструкцията p_2 на всяко от работещите копия да се изпълни след като инструкция p_1 е завършила изпълнението си в поне 3 работещи копия.

Упътване: Освен семафори, ползвайте и брояч.

Зад. 60 Опишете реализацията на комуникационна тръба (pipe) чрез семафори. Предполагаме, че тръбата може да съхранява до n байта, подредени в обикновена опашка. Тръбата се ползва от няколко паралелно работещи изпращачи/получатели на байтове. Процесите изпращачи слагат байтове в края на опашката, получателите четат байтове от началото на опашката.