Поправителен изпит (задачи), част I, за курса “Операционни системи”, СУ, ФМИ, 26.08.2017 г.

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

Задача	1 (20)	2 (20)	3 (20)	4 (20)
получени точки				

Забележка: За удобство приемаме, че в имената на файловете и директориите няма специални символи. Ако имате нужда, можете да ползвате временни файлове.

Задача 1. Напишете скрипт, който получава два задължителни позиционни параметъра – директория и низ. Сред файловете в директорията би могло да има такива, чиито имена спазват следната 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
```

Задача 2. 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 директориите 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
```

Задача 3. 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`.

Задача 4. 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`.

Примерни решения

Задача 1.

```
#!/bin/dash

if [ "$#" -ne 2 ]; then
    echo npar 1>&2
    exit 1
fi

if [ ! -d "$1" ]; then
    echo ndir 1>&2
    exit 1
fi

LANG=C
export LANG

fullnames=$(mktemp)
find -- "$1" -mindepth 1 -maxdepth 1 -type f -name "vmlinuz-*.*.*-${2}" > "${fullnames}"
if [ ! -s "${fullnames}" ]; then
       rm -- "${fullnames}"
       exit 1
fi

fnames=$(mktemp)
xargs -n1 -- basename -- < "${fullnames}" > "${fnames}"
rm -- "${fullnames}"

if egrep -qve '^vmlinuz-[0-9]+\.[0-9]+\.[0-9]+-' -- "${fnames}"; then
    rm -- "${fnames}"
    echo invalid filenames 1>&2
    exit 1
fi

ver="$(cut -d- -f2 -- "${fnames}" | sort -n -t. -k1,1 -k2,2 -k3,3 | tail -n1)"
pat="$(printf '%s' "${ver}" | sed -e 's/\./\\./g')"
egrep -e "^vmlinuz-${pat}-" -- "${fnames}"

rm -- "${fnames}"
```

Задача 1. Алтернативно решение

```
#!/bin/dash

if [ $# -ne 2 ]; then echo npar 1>&2 ; exit 1; fi

if [ ! -d $1 ]; then echo ndir 1>&2 ; exit 1; fi

longname=$(find -- "$1" -mindepth 1 -maxdepth 1 -type f -name "vmlinuz-*.*.*-${2}" \
    | sort -V | tail -n1)

[ -z "${longname}" ] || basename "${longname}"
```

Задача 2.

```
#!/bin/dash
```

```
find_bad_users() {
    cat /etc/passwd | while IFS=: read u h i g e h s; do
        [ "$i" != '0' ] || continue

        if [ -d "$h" ]; then
            oid=$(stat -c '%u' -- "$h")
            ow=$(stat -c '%A' -- "$h" | cut -c 3)
        fi

        if [ ! -d "$h" ] || [ "$oid" -ne "$i" ] || [ "$ow" != 'w' ]; then

            user_rss=$(ps -u "$u" -o rss= | awk '{ sum += $1 } END { print sum }')
            [ -n "$user_rss" ] || continue

            if [ "$user_rss" -gt "$root_rss" ]; then
                echo "$u"
            fi
        fi
    done
}

if [ "$(id -u)" != '0' ]; then
    echo nroot 1>&2
    exit 1
fi

root_rss=$(ps -u root -o rss=  | awk '{ sum += $1 } END { print sum }')

baduser="$(find_bad_users)"
if [ -n "$baduser" ]; then

    for u in $baduser; do
        pkill -TERM -u "$u"
    done

    sleep 5

    for u in $baduser; do
        pkill -KILL -u "$u"
    done
fi
```

Задача 3. За двете искани в условието синхронизации използваме два семафора – `t1` и `t2`, инициализираме ги с блокиращо начално състояние:

```
semaphore t1,t2
t1.init(0)
t2.init(0)
```

Добавяме в кода на процесите `P` и `Q` синхронизиращи инструкции:

```
process P                process Q
  p_1                      q_1
  t1.signal()              t2.signal()
  t2.wait()                t1.wait()
  p_2                      q_2
```

Инструкцията `q_2` ще се изпълни, след като процесът `Q` премине бариерата `t1.wait()`. Това се случва след изпълнението от `P` на ред `t1.signal()`, който следва инструкция `p_1`.

Аналогично, инструкцията `p_2` ще се изпълни след изпълнението на ред `t2.signal()`, който следва инструкция `q_1`.

Решението на задачата осигурява среща във времето (rendezvous) на двата процеса. Важен е редът на извикване на инструкциите, управляващи семафорите. Ако го обърнем, получаваме класически пример за deadlock:

```
process P                 process Q
  p_1                       q_1
  t2.wait()                 t1.wait()
  t1.signal()               t2.signal()
  p_2                       q_2
```

Задача 4. Използваме два семафора – `s_p` и `s_q`, инициализираме ги така:

```
semaphore s_p, s_q
s_p.init(1)
s_q.init(0)
```

Добавяме в кода на процесите `P` и `Q` синхронизиращи инструкции:

```
process P                 process Q
  s_p.wait()                s_q.wait()
  p_1                       q_1
  p_2                       q_2
  s_q.signal()              s_p.signal()
```