Второ контролно за курса 'Операционни системи', СУ, ФМИ, 13.05.2017 г.

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

Задача	1 (30)	2 (30)	3 (30)	4 (40)	Общо (130)
получени точки					

Забележка: За всички задачи, в имената на файловете и директориите няма специални символи.

Задача 1. 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.

Задача 2. 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
  ```

Задача 3. 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`

Задача 4. 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.

Примерни решения

Задача 1.

```
#!/bin/bash
check_dir() {
    if [ ! -d "$1" ]; then
        echo "$1 not a dir"
        exit 2
    fi
}

case $# in
    1)
        check_dir "$1"
        find -L $1 -type l
        ;;
    2)
        check_dir "$1"
        for i in $(find "$1" -type f); do
            if [ $(stat -c '%h' "${i}") -ge $2 ]; then
                echo "${i}"
            fi
        done
        ;;
    *)
        echo "usage: $0 <dirname> [<number>]"
        exit 1
esac
```

Задача 2.

```
#!/bin/bash
check_dir() {
    if [ ! -d "$1" ]; then
        echo "$1 not a dir"
        exit 3
    fi
}

if [ $# -ne 3 ]; then
    echo "usage: $0 <dirname> <dirname> <string>"
    exit 1
fi

if [ $(id -u) -ne 0 ]; then
    echo "must be run as root"
    exit 2
fi

check_dir "${1}"
check_dir "${2}"

if [ $(find "${2}" -type f | wc -l) -ne 0 ]; then
    echo "${2} contains files"
    exit 4
fi

SRC=$(dirname "${1}")"/"$(basename "${1}")
```

```
DST=$(dirname "${2}")"/"$(basename "${2}")
ABC="$3"

for i in $(find "${SRC}" -type f -name "*${ABC}*"); do
    DSTFILE=$(echo ${i} | sed -e "s|${SRC}|${DST}|")
    mkdir -p $(dirname $DSTFILE)
    mv ${i} ${DSTFILE}
done
```

Задача 3.

```
#!/bin/bash
if [ $(id -u) -ne 0 ]; then
    echo "must be run as root"
    exit 1
fi

for U in $(ps -e -o user | tail -n +2 | sort | uniq); do
    TOTAL_RSS=0
    PS_COUNT=0

    while read CPID CRSS; do
        PS_COUNT=$(expr ${PS_COUNT} + 1)
        TOTAL_RSS=$(expr ${TOTAL_RSS} + ${CRSS})
        MAX_RSS=${CRSS}
        MAX_RSS_PID=${CPID}
    done < <(ps -u ${U} -o pid,rss | tail -n +2 | sort -n -k 2)

    if [ ${PS_COUNT} -gt 0 ]; then
        AVGRSS=$(expr ${TOTAL_RSS} / ${PS_COUNT})
        echo "${U} ${PS_COUNT} ${TOTAL_RSS}"

        if [ ${MAX_RSS} -gt $(expr ${AVGRSS} "*" 2) ]; then
            echo -e "\tpid ${MAX_RSS_PID} has ${MAX_RSS}, will terminate"
            kill -s SIGTERM ${MAX_RSS_PID}
            sleep 2
            kill -s SIGKILL ${MAX_RSS_PID}
        fi
    fi
done
```

Задача 4. Първо забелязаме, че семафорът `m` се ползва само от `P` в ролята на mutex. В неговата опашка може да има само копия на `P` и само едно работещо копие може да намалява/увеличава брояча синхронизирано с блокирането/освобождаването на семафора `e`.

Увеличаването на `cnt` става преди критичната секция на `P`, а намалянето след нея. Ако не вървят никакви копия на `Q`, лесно се убеждаваме, че могат да се изпълняват произволен брой критични секции на `P`, като броячът съвпада с броя на паралелно изпълняваните критични секции. Така отговорът на въпрос (б) е ДА.

Заемането на семафора `e` в `P` става точно когато `cnt` променя стойността си от 0 в 1. Освобождаването става точно когато `cnt` променя стойността си от 1 в 0.

Тъй като при инициализацията броячът на `e` е 1, а употребата му и в двата вида процеси започва със заемане и завършва с освобождаване, само едно копие от двата типа ще може да премине `e.wait()`. Разглеждаме два случая:

(А) Процесът `Q` преминава. Тогава ще се изпълни критичната му секция, но само от това копие. Останалите копия на `Q` ще бъдат приспани от първата си инструкция. Следователно отговорът на въпрос (в) е НЕ.

Ако версия на `P` пробва `e.wait()`, тя също ще бъде приспана. Това ще стане точно когато `cnt` променя стойността си от 0 в 1, тоест не се изпълняват критични секции на `P`. В момента на

приспиване и мутекса `m` е блокиран. Това обстоятелство ще блокира всички опити на други копия на `P` да преминат `m`. В този случай в опашката на семафора `e` има точно едно копие на `P`.

(B) Процесът `P` преминава. Ще започне изпълнение на неговата критична секция и евентуално на други копия на `P`, докато `cnt>0`. През този период всички копия на `Q` ще бъдат приспани от първата си инструкция. Когато `cnt` намалее до 0, никое копие не изпълнява критична секция.

От двата разгледани случая следва, че в един момент могат да се изпълняват няколко критични секции на `P` или една критична секция на `Q`. Следователно отговорът на въпрос (а) е НЕ.

В описаната схема няма условия за deadlock. `Q` не може да инициира deadlock, тъй като ползва само един ресурс. `P` също не може поради реда на заемане на ресурсите (първо заема семафора `m`, после `e`).

В описаната схема има условия за гладуване (starvation) на процес `Q`. Нека критичната секция на `P` се изпълнява бавно и `Q` започва работа след `P`. Ще започне изпълнение на критична секция на `P` и ако постоянно започват работа нови копия, броячът `cnt` може да остане положителен неограничено време. Така `Q` ще бъде приспан неограничено дълго.