Operating Systems

Started on	Monday, 24 April 2023, 6:13 PM
State	Finished
Completed on	Monday, 24 April 2023, 7:08 PM
Time taken	55 mins
Grade	**8.35** out of 15.00 (**55.67**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Odrezati želimo zadnje tri znake podanega niza. Katera rešitev je pravilna? (Možnih je več odgovorov)

We want to cut off the last three characters of the given string. Which are the correct solutions? (Multiple answers are possible)

- [✓ a.] `a=${#niz}; niz=${niz::a-3}` ✓
- b. `niz=${niz:(-3)}`
- [✓ c.] `a=${#niz}; a=$(($a-3)); niz=${niz:0:$a}` ✓
- d. `niz=${niz:-3}`
- [✓ e.] `a=${#niz}; niz=${niz::$a-3}` ✓
- [✓ f.] `a=${#niz}; niz=${niz:0:a-3}` ✓
- g. `niz=${niz2%.*}`
- [✓ h.] `a=${#niz}; niz=${niz:0:$a-3}` ✓

Vaš odgovor je pravilen.

The correct answers are:
`a=${#niz}; niz=${niz::a-3}`

,
`a=${#niz}; niz=${niz::$a-3}`

,
`a=${#niz}; niz=${niz:0:$a-3}`

,
`a=${#niz}; niz=${niz:0:a-3}`

,
`a=${#niz}; a=$(($a-3)); niz=${niz:0:$a}`

Question **2**

Partially correct

Mark 0.75 out of 1.00

Flag question

Kako vse bi lahko implementirali neskončno zanko v Bashu? Izberite izmed ponujenih primerov.
Možen je en, dva ali več odgovorov.

How could we implement an infinite loop in Bash? Choose from the examples provided.
One, two or more answers are possible.

- [✓ a.] while [1 -ne 0]; do ✓
- [✓ b.] while true; do ✓
- [✓ c.] while :; do ✓
- [✓ d.] until false; do ✓
- e. while 1; do
- f. until true; do
- g. while false; do
- [✓ h.] for ((i=0; i>0; i++)); do ✗

Vaš odgovor je delno pravilen.

You have selected too many options.
The correct answers are:
while true; do,

while :; do,

while [1 -ne 0]; do,

until false; do

Question **3**

Partially correct

Mark 0.75 out of 1.00

Flag question

Kako lahko podamo parametre funkciji v bashu?
Možen je en odgovor.

How can I pass parameters to a function in bash?
There is one possible answer.

- a. `v podlupini $()`
  `in subscript $()`
- [✓ b.] `function sestej(a, b)` ✗
- c. `preko ukaza return`

  with **return** command
- [✓ d.] `preko argumentov in potem uporabimo $1, $2` ✓

  `with arguments and then use $1, $2`
- e. `preko ukaza echo`

  with **echo** command

Vaš odgovor je delno pravilen.

You have selected too many options.
The correct answer is:
`preko argumentov in potem uporabimo $1, $2`

`with arguments and then use $1, $2`

Question **4**

Partially correct

Mark 0.50 out of 1.00

Flag question

Označi katere trditve držijo?
Možen je en, dva ali več odgovorov.

Which statements are correct?
One, two or more answers are possible.

- [✓ a.] Funkcije lahko kličemo s parametri, na katere se znotraj funkcije sklicujemo z ✓
  `$1, $2,…`

  Functions can be called with parameters that are referenced within the function by
  `$1, $2,…`
- [✓ b.] Funkcija mora biti definirane preden jo lahko kličemo. ✓

  A function must be defined before it can be called.
- c. Funkcijo pokličeš z njenim imenom kot
  `ime_funkcije`

  brez oklepajev.

without parentheses.

- [ ] d. Spremenljivke, ki so definirane znotraj funkcij, so globalne.
  Variables defined within functions are global.

Vaš odgovor je delno pravilen.

You have correctly selected 2.
The correct answers are:
Funkcija mora biti definirane preden jo lahko kličemo.

A function must be defined before it can be called.,

Funkcije lahko kličemo s parametri, na katere se znotraj funkcije sklicujemo z

```
$1, $2,…
```

Functions can be called with parameters that are referenced within the function by

```
$1, $2,…
```

,
Funkcijo pokličeš z njenim imenom kot

```
ime_funkcije
```

brez oklepajev.
You call a function by its name as

```
function_name
```

without parentheses.
,
Spremenljivke, ki so definirane znotraj funkcij, so globalne.

Variables defined within functions are global.

Question **5**
Partially correct
Mark 0.75 out of 1.00
Flag question

Kateri ukaz shrani rezultat iskanja v `rezultat.txt`, napake pa v `napake.txt`
Možen je en, dva ali več odgovorov.

Which command saves the search result in `rezultat.txt` and the errors in `napake.txt`
One, two or more answers are possible.

- [x] a. find -name "*os*" > rezultat.txt 2>napake.txt✓
- [x] b. find -name "*os*" 1>rezultat.txt 2>napake.txt✓
- [ ] c. find -name "*os*" &>rezultat.txt 2>napake.txt
- [ ] d. find -name "*os*" >> rezultat.txt 1>napake.txt
- [x] e. find -name "*os*" 2>napake.txt 1>rezultat.txt✓

Vaš odgovor je delno pravilen.

You have correctly selected 3.
The correct answers are:
find -name "*os*" > rezultat.txt 2>napake.txt,

find -name "*os*" 1>rezultat.txt 2>napake.txt,

find -name "*os*" &>rezultat.txt 2>napake.txt,

find -name "*os*" 2>napake.txt 1>rezultat.txt

Question **6**
Partially correct
Mark 0.80 out of 1.00
Flag question

Zapišite zaporedje ukazov v Bashu, ki ustreza prikazanemu diagramu:

Write down the sequence of commands in Bash that correspond to the following diagram:

diagram

Answer: cat data.txt 2>error.txt | tail -5 2>echo | sort 1>result.txt 2>error_2.txt ✖

The correct answer is: cat data.txt 2> error.txt | tail -5 | sort > result.txt 2> error_2.txt

Comment:

Question **7**
Correct
Mark 1.00 out of 1.00
Flag question

Predpostavimo, da se v imeniku /home/student nahaja datoteka "seznam_datotek.txt", ki vsebuje stolpec z imeni datotek (po ena vrstica za vsako ime datoteke).
Zapišite zaporedje ukazov, ki za vse datoteke imenovane v "seznam_datotek.txt" poženejo ukaz ls in blok pravic datotek zapišejo v datoteko "pravice.txt" (v istem imeniku).

Assume that there is a file "seznam_datotek.txt" in the /home/student directory, which contains a column with filenames (one line for each filename). Write a sequence of commands that execute the ls command for all files named in "seznam_datotek.txt" and write the file permissions block to the "pravice.txt" file (in the same directory).

Primer vsebine datoteke pravice.txt:

```
drwxr-xr-x
-rw-rw-r--
-rw-r--r--
```

Answer: cat seznam_datotek.txt | xargs ls -l | cut -d" " -f1 > pravice.txt ✖

The correct answer is: cat seznam_datotek.txt | xargs ls -l | cut -d " " -f1 >pravice.txt

Comment:

Question **8**
Correct

Zapišite zaporedje ukazov, ki prebere prvih 100 vrstic generatorja psevdonaključnih števil in izpiše koliko besed vsebuje generiran niz.

Write a sequence of commands that reads the first 100 lines of the pseudorandom number generator and prints how many words the generated string contains.

Answer: `cat /dev/urandom | tr -d -c "0-9\n" | head -n 100 | wc -w` ✗

The correct answer is: cat /dev/urandom | head -100 | wc -w

Comment:

Question **9**

Correct

Mark 1.00 out of 1.00

Flag question

Prešteti želimo število datotek (ne glede na vrsto) v trenutnem imeniku. Zapišite zaporedje ukazov, s katerim to dosežemo.

We want to count the number of files (of any type) in the current directory. Write a sequence of commands to achieve this.

Primer izpisa, če je 5 datotek v imeniku:
Example of print if there are 5 files in the directory:

```
5
```

Answer: `ls | wc -l` ✓

The correct answer is: ls | wc -l

Question **10**

Partially correct

Mark 0.80 out of 1.00

Flag question

V spremenljivki `niz` je shranjeno ime uporabnika. Zapišite zaporedje ukazov, s katerim preverite če ta uporabnik obstaja. Če uporabnik obstaja potem spremenljivki dodajte `" obstaja"`, drugače `" ne obstaja"`, in spremenljivko prepišite. **V obeh primerih v terminal izpišite vrednost posodobljene spremenljivke.** Ukaz naj v terminal ne izpisuje drugih rezultatov.

The user's username is stored in variable `niz`. Write a sequence of commands to check whether this user exists. If the user exists then add `" obstaja"` to the variable, otherwise `" ne obstaja"` and overwrite the variable. **In both cases output the value of the updated variable to the terminal.** The command should not output any other results to the terminal.

Primer izpisa za uporabnika `student`, ki obstaja:
Example output for existing user `student`:

```
student obstaja
```

Primer izpisa za uporabnika `gost`, ki ne obstaja:
Example output for non-existing user `gost`:

```
gost ne obstaja
```

Answer: `cat /etc/passwd | grep -a $niz >/dev/null && echo "$niz obstaja" || echo "$niz n` ✗

The correct answer is: id -u $niz &> /dev/null && { niz=$niz" obstaja"; echo $niz; } || { niz=$niz" ne obstaja"; echo $niz; }

Comment:

Question **11**

Complete

Mark 0.00 out of 5.00

Flag question

Napiši bash skripto, ki:

1. V ozadju požene ukaz xeyes.
2. V konzolo izpiše število odprtih datotečnih deskriptorjev procesa, v katerem se skripta izvaja.
3. V konzolo izpiše število vseh pojavitev datumov v ISO formatu (YYYY-MM-DD) v navodilih nevgrajenega ukaza, ki ga prejme kot prvi argument. Če argument ni podan, naj pojavitve išče v navodilih ukaza "date". Kot primer: zapis "2004-02-14" ustreza ISO formatu, prav tako tudi "0904-03-07", sledeči zapisi pa ne: "2004/02/14", "14-02-2004", "2004-2-14", "904-3-7", "04-02-14", "22004-02-14", "2004-02-142". Da bo naloga lepše rešljiva, lahko kot veljavne dneve in mesece upoštevaš 00-99, kot veljavna leta pa 0000-9999. Pozoren bodi na to, da je lahko v posamezni vrstici več datumov - v tem primeru šteješ vsakega posebej.
4. V neskončni zanki spi po 1 sekundo naenkrat.
5. V primeru, da skripta kadarkoli med izvajanjem prejme signal SIGUSR1 ali SIGUSR2, naj v konozolo izpiše "Ubil si me", pošlje procesu xeyes, ki ga je skripta pognala, signal SIGKILL in se zaključi z izhodnim statusom 0. Pri tem upoštevaj, da na sistemu lahko tečejo še kakšni drugi procesi xeyes (katerim signala SIGKILL ne pošiljaj).

Pri tem bodi pozoren na sledeče:

- skripta naj bo napisana v taki obliki, da jo je možno pognati (torej naj ne vsebuje sintaktičnih nepravilnosti ali ukazov namenjenih izvajanju izven skripte);
- rešitve posameznih podnalog so lahko sestavljene iz več ukazov, ne nujno le enega;
- vrstni red izpisov in izvajanja rešitev posameznih podnalog naj sledi vrstnemu redu navodil, razen kjer je potrebno vrstni red zamenjati za pravilno delovanje;
- skripta naj nima stranskih učinkov: to pomeni, naj skripta (razen, kjer je to potrebno za rešitev naloge) ne spreminja pravic, ne piše in briše po disku, ne izpisuje stvari v konzolo, ne izvaja ukazov kot superuporabnik (angl. superuser, root), ipd.;
- če skripta uporablja ukaze, ki jih na vajah nismo uporabljali (npr. awk in sed), se pričakuje, da boste njihovo uporabo znali po potrebi razložiti in prirediti novim primerom uporabe;
- če skripto kopiraš iz virtualke, jo skopiraj pravi čas (ne zadnjih 15 sekund kviza) in pazi, da res skopiraš celotno zadevo (predvsem pazi, da se predolge vrstice ne porežejo).

Write a bash script that:

1. Runs xeyes in the background.
2. Prints the number of file descriptors opened in the process, in which the script is running.
3. Prints the number of all ISO-format-compliant (YYYY-MM-DD) dates in the instruction manual of the external (i.e. not built-in) command passed as the first argument. If the argument is not passed, it should search in the instructions of the "date" command. E.g.: "2004-02-14" is ISO-compliant, as is "0904-03-07", while the following are not: "2004/02/14", "14-02-2004", "2004-2-14", "904-3-7", "04-02-14", "22004-02-14", "2004-02-142". For the sake of simplicity, you can treat 00-99 as valid days and months, and 0000-9999 as valid years. Keep in mind that there can be multiple dates in a single line, in which case you should count each of them separately.
4. Sleeps in 1-second intervals in an infinite loop.
5. If the script receives the SIGUSR1 or SIGUSR2 signal at any point during execution, it should print "Ubil si me", send the signal SIGKILL to the xeyes process started by this script, and terminate with the exit status 0. Note that there may be other xeyes processes present on the system (which shouldn't be sent the SIGKILL signal).

Keep in mind the following:

- the script should be written in a way that it can be executed as such (i.e. it should be syntactically correct and should not contain commands to be executed outside the script);
- the solutions of each subproblem can be comprised of several commands, not necessarily just one;
- the order of the printouts and execution of the subproblem solutions should be the same as the order in the instructions, except where the order needs to be changed for the solutions to work correctly;
- the script should not have side-effects, in particular this means that, unless necessary, the script should not: change file permissions, write to disk or delete files, print to console, execute commands as superuser (root user), etc.;
- if the script uses commands we didn't mention in our lab sessions (such as awk or sed), you will be expected to be able to explain their use, and adapt them to new use cases;
- if you're copying the script from the virtual machine, make sure to copy it on time (not with 15 seconds of quiz time remaining) and make sure you copy the entire script (take special care that lines that are too long don't get cut off).

```
#!/bin/bash
test -z 1 && spremenljivka=. || spremenljivka= 1 && spremenljivka=. || spremenljivka= 1
ln -s | cut -d" " -f2
```

Odgovor oddan.

Comment:

Finish review

Jump to...

2. kolokvij (skupina ob 19h) ►

Quiz navigation

Vprašanja

1	2	3	4	5	6	7	8	9	10
✓	●	●	●	●	●	✓	✓	✓	●

Programerska naloga

11
🟥

Show one page at a time

Finish review