

Řídícím operátorem není

- a) //
- b) ||
- c) &&
- d) ;;

Znak, který následuje za \ se

- a) nepovažuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
- b) nepovažuje za řídící. Je-li tím znakem "nový řádek", nepokračuje se na dalším řádku.
- c) považuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
- d) považuje za řídící. Je-li tím znakem "nový řádek", nepokračuje se na dalším řádku.

Znaky uzavřené do dvojice apostrofů ('...') ztrácejí svůj řídící význam s výjimkou

- a) ' (apostrof)
- b) \$, ` (obrácený apostrof), \ (následuje-li \$, `, ", \, nový řádek)
- c) ' (apostrof), \
- d) ` (obrácený apostrof)

Znaky uzavřené do dvojice uvozovek ("...") ztrácejí svůj řídící význam s výjimkou

- a) ' (apostrof)
- b) \$, ` (obrácený apostrof), \ (následuje-li \$, `, ", \, nový řádek)
- c) ' (apostrof), \
- d) ` (obrácený apostrof)

Adresář se jménem "Ferda mravenec" (má ve jméně mezeru) nevytvořím příkazem

- a) mkdir "Ferda mravenec"
- b) mkdir 'Ferda mravenec'
- c) mkdir `Ferda mravenec`
- d) mkdir Ferda\ mravenec
- e) mkdir Ferda" "mravenec

Co se předá na standardní výstup po spuštění příkazu **echo \**

- a) Bad filename
- b) echo \
- c) \
- d) "
- e) '

Co se předá na standardní výstup po spuštění příkazu **echo \\\"'**

- a) \"'
- b) \"\\\"'
- c) \\\"'
- d) \\\"'

Příkaz při ukončení vrací ukončovací kód. Jaký ukončovací kód je?

- a) číselný; 0 znamená úspěšné ukončení, nenulové znamená ukončení s chybou
- b) číselný; 1 znamená úspěšné ukončení, 0 znamená ukončení s chybou
- c) číselný; nenulový znamená úspěšné ukončení, 0 znamená ukončení s chybou
- d) textový; 'ok' znamená úspěšné ukončení, prázdný řetězec znamená ukončení s chybou
- e) textový; prázdný řetězec znamená úspěšné ukončení, neprázdný řetězec znamená ukončení s chybou

Znak \ má následující význam:

- a) oddělovač adresářů a jména souboru od adresáře v cestě
- b) zapnutí pokračovacího řádku
- c) označení control znaku
- d) zrušení řídícího charakteru následujícího znaku

Znak ' (apostrof) má následující význam:

- a) všechny znaky v řetězci uzavřeném do dvojice '...' ztrácí řídící význam
- b) příkaz uzavřený do dvojice '...' se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
- c) znaky v řetězci uzavřeném do dvojice '...' ztrácí řídící význam vyjma řídících znaků \$ \
- d) označuje control znak

Znak " má následující význam:

- a) všechny znaky v řetězci uzavřeném do dvojice "..." ztrácí řídící význam
- b) příkaz uzavřený do dvojice "..." se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
- c) znaky v řetězci uzavřeném do dvojice "..." ztrácí řídící význam vyjma řídících znaků \$ \
- d) označuje control znak

Kolik obrácených lomítek a kolik apostrofů předá na standardní výstup příkaz **echo \"'\"'**

- a) 2 a 1
- b) 2 a 2
- c) 3 a 2
- d) 3 a 3
- e) 4 a 4

Který z příkazů nesmaže soubor x?

- a) rm x
- b) rm 'x'
- c) rm "x"
- d) rm `x`
- e) rm \x

Příkaz **sort > soubor1 < soubor2**

- a) čte standardní vstup ze souboru **soubor2** a standardní výstup zapisuje do souboru **soubor1**
- b) čte standardní vstup ze souboru **soubor1** a standardní výstup zapisuje do souboru **soubor2**
- c) čte standardní vstup ze souboru **soubor2** a standardní chybový výstup zapisuje do souboru **soubor1**
- d) čte standardní vstup ze souboru **soubor1** a standardní chybový výstup zapisuje do souboru **soubor2**

Operátory přesměrování jsou

- a) >&, >, >>, <<-, <<
- b) <<, >&, <->, <&
- c) <, >, >|, <=, >>>
- d) >>, <<, <\$, <, >, \$>

Který z příkazů nezkopíruje soubor 'a' do souboru 'b'?

- a) cp a b
- b) cat a > b
- c) cat < a > b
- d) cat a|cat > b
- e) cp < a > b

Jaké je korektní a smysluplné přesměrování vstupu?

- a) ls < adresar
- b) cd < adresar
- c) grep Brno < mesta
- d) ls > adresar

Jaké je korektní a smysluplné přesměrování výstupu?

- a) echo toto se mi líbí > soubor
- b) cd adresar > soubor
- c) mv a > b
- d) grep Brno < města

Soubor 'a' před provedením příkazu **ls > a**

- a) musí existovat
- b) nesmí existovat
- c) může existovat

Soubor 'a' před provedením příkazu **sort < a**

- a) musí existovat
- b) nesmí existovat
- c) může existovat

Po provedení příkazu

**echo je > je; rm -rf neni; ls je neni > a 2> b**

- a) bude soubor 'a' větší než soubor 'b'
- b) bude soubor 'b' větší než soubor 'a'
- c) budou soubory 'a' a 'b' stejně velké
- d) soubor 'b' nebude existovat nebo bude prázdný
- e) soubor 'a' nebude existovat nebo bude prázdný

Po provedení příkazu **echo a > b > c**

- a) bude soubor 'b' prázdný a soubor 'c' neprázdný
- b) budou soubory 'b' a 'c' stejně velké
- c) bude soubor 'b' neprázdný a soubor 'c' prázdný
- d) soubor 'b' nebude existovat
- e) soubor 'a' se vyprázdní

Po provedení příkazu

**echo b > s; echo a >> s; sort < s > s**

- a) bude soubor 's' prázdný
- b) bude soubor 's' obsahovat řádky v pořadí 'a' a 'b'
- c) bude soubor 's' obsahovat řádky v pořadí 'b' a 'a'
- d) nebude soubor 's' existovat

Po provedení příkazů

**\$ echo 'mravenec' > a; ls -l a**

**-rw-r--r-- 1 brandejs staff 9 dub 8 23:23 a**

**\$ echo ferda >> a**

bude mít soubor 'a' velikost

- a) 5
- b) 6
- c) 13
- d) 14
- e) 15

Posloupnost příkazů

**cat && ukazka**

**ls -l ukazka**

**ukazka**

předá na standardní výstup

- a) obsah souboru 'ukazka', výstup příkazu 'ls -l ukazka', spustí se program 'ukazka' z běžného adresáře
- b) text 'ls -l ukazka'
- c) text 'ukazka'
- d) nepředá se nic
- e) předá se výstup programu 'ukazka'

Spojení příkazů do kolony je

- a) ls|grep txt|sort
- b) ls;grep txt;sort
- c) ls&grep txt&sort
- d) ls&&grep txt&&sort

Návratový kód seznamu **rm -rf adr;mkdir adr;**

**touch adr/so;ls adr/so|sort|grep soubor** bude

- a) 0
- b) nenulové kladné číslo
- c) nenulové záporné číslo
- d) prázdný řetězec
- e) text s chybovým hlášením

Návratový kód seznamu **rm -rf adr;**

**touch adr/soubor;echo adr/soubor|sort|grep soubor** bude

- a) 0
- b) nenulové kladné číslo
- c) nenulové záporné číslo
- d) prázdný řetězec
- e) text s chybovým hlášením

Příkaz spustíme na pozadí, když

- a) za příkaz napíšeme &
- b) před příkaz napíšeme &
- c) za příkaz napíšeme \$
- d) před příkaz napíšeme \$

Pro příkaz spuštěný na pozadí neplatí

- a) lze ho ukončit speciálním znakem INTR
- b) lze ho ukončit příkazem kill
- c) standardní vstup se napojí na /dev/null
- d) byl spuštěn pomocí oddělovače &

Kam je napojen standardní vstup procesu spuštěného na pozadí?

- a) /dev/null
- b) /dev/tty
- c) /dev/zero
- d) /dev/console

Který z příkazů předá na standardní výstup nejprve řádek obsahující 'b' a potom řádek obsahující 'a'?

- a) sleep 2;echo a&sleep 1;echo b
- b) sleep 1;echo a&sleep 2;echo b
- c) (sleep 1;echo a)&sleep 2;echo b
- d) (sleep 2;echo a)&sleep 1;echo b

Spustím příkaz **(sleep 10;echo a)&sleep 10;echo b** a po spuštění stisknu zvláštní znak INTR (^C). Co se předá na standardní výstup dříve?

- a) a
- b) b
- c) předají se řádky 'a' i 'b' ve stejný čas v náhodném pořadí
- d) nepředá se nic

Provedete posloupnost příkazů/operací:

**cat > a&cat > b**

**ahoj**

**^D**

Který soubor bude větší?

- a) a
- b) b
- c) soubory 'a' i 'b' budou stejně velké a neprázdné
- d) soubory 'a' i 'b' budou stejně velké a prázdné
- e) jedno ^D nestačí, musí se stisknout dvakrát

Jakým způsobem nelze úlohu dostat pod správu Job Controllu?

- a) příkazem fg
- b) příkazem bg
- c) spuštěním s &
- d) speciálním znakem SUSP (^Z)

Úloha běžící na pozadí se přesune na popředí příkazem

- a) lg
- b) kill
- c) nohup
- d) fg
- e) žádná jiná odpověď není správná

Pro seznam neplatí

- a) je to posloupnost žádného nebo více příkazů
- b) příkazy jsou odděleny novým řádkem nebo středníkem (;) nebo ampersandem (&)
- c) shell provádí příkazy v pořadí, v jakém jsou zapsány
- d) návratový kód seznamu je návratový kód prvního prováděného procesu
- e) pokud příkaz končí ampersandem (&), ihned se zahájí provádění následujícího příkazu

Oddělením dvou příkazů oddělovačem &&

- a) spustíme první příkaz na popředí a druhý příkaz na pozadí
- b) spustíme první příkaz na pozadí a druhý příkaz na popředí
- c) provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
- d) provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód

K čemu slouží oddělovač && mezi dvěma příkazy?

- a) druhý příkaz se provede, pokud je návratový kód prvního nenulový
- b) druhý příkaz se provede, pokud je návratový kód prvního nula
- c) druhý příkaz se spustí hned po spuštění prvního
- d) druhý příkaz se spustí na pozadí hned po spuštění prvního

Oddělením dvou příkazů oddělovačem ||

- a) spustíme první příkaz na popředí a druhý příkaz na pozadí
- b) spustíme první příkaz na pozadí a druhý příkaz na popředí
- c) provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
- d) provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód

K čemu slouží oddělovač || mezi příkazy?

- a) druhý příkaz se provede, pokud je návratový kód prvního nenulový
- b) druhý příkaz se provede, pokud je návratový kód prvního nula
- c) druhý příkaz se spustí hned po spuštění prvního
- d) druhý příkaz se spustí na pozadí hned po spuštění prvního

Příkazem **ls adresar 2>/dev/null || mkdir adresar**

- a) vytvoříme adresář, pokud neexistuje
- b) vytvoříme adresář, pokud příkaz ls vrátil nulový návratový kód
- c) vypíšeme vždy obsah adresáře a vytvoříme nový adresář
- d) vypíšeme obsah adresáře

Vyberte nesprávné tvrzení o seznamu **cd zkusto&&rm \***

- a) pokud adresář zkusto nebude existovat, zruší se všechny soubory běžného adresáře
- b) první příkaz bude mít návratovou hodnotu 0, pokud existuje přístupný adresář zkusto
- c) pokud adresář zkusto bude existovat, zruší se v něm všechny soubory
- d) pokud první příkaz bude mít návratovou hodnotu 1, tak se druhý neprovede

Příkaz **rm `cat files.txt`**

- a) vymaže všechny soubory, které jsou uvedeny v souboru files.txt
- b) vymaže soubor files.txt
- c) vymaže soubor cat i soubor files.txt
- d) vymaže soubor files.txt a také soubory, které jsou v něm uvedeny

Příkaz **rm cat files.txt**

- a) vymaže všechny soubory, které jsou uvedeny v souboru files.txt
- b) vymaže soubor files.txt
- c) vymaže soubor cat i soubor files.txt
- d) vymaže soubor files.txt a také soubory, které jsou v něm uvedeny

Příkaz **rm -rf \*;touch a;echo 'ls' `ls`** předá na standardní výstup

- a) ls a
- b) a ls
- c) obsah souboru 'a'
- d) 'ls' ls
- e) ls `ls`

Příkaz **echo a > seznam;rm 'seznam'** smaže soubor

- a) a
- b) seznam
- c) příkaz je chybný
- d) 'a' i 'seznam'

Příkaz **echo a > 'seznam';rm `cat seznam`** smaže soubor

- a) a
- b) seznam
- c) příkaz je chybný
- d) 'a' i 'seznam'

Příkaz **echo a > 'seznam';rm `echo seznam`** smaže soubor

- a) a
- b) seznam
- c) příkaz je chybný
- d) 'a' i 'seznam'

Příkaz **echo a > `seznam`;rm 'cat seznam'** smaže soubor

- a) a
- b) seznam
- c) příkaz je chybný
- d) 'a' i 'seznam'

Proměnná se v shellu "deklaruje" příkazem

- a) var ADRESAR=/tmp
- b) ADRESAR=/tmp
- c) \$ADRESAR=/tmp
- d) strings \$ADRESAR /tmp

Jak vytvořím proměnou ADRESAR obsahující slovo EMPTY

- a) ADRESAR=EMPTY
- b) ADRESAR:=EMPTY
- c) \$ADRESAR=EMPTY
- d) touch ADRESAR < EMPTY

Zadali jsme **velikost=maxi** Jak vypíšeme řetězec "maxipes"?

- a) echo \$velikost."pes"
- b) echo \${velikost}pes
- c) echo \$velikostpes
- d) echo \${velikost}pes

Deklarace proměnné spolu s jejím naplněním se provádí

- a) jmeno\_promenne=[hodnota]
- b) jmeno\_promenne:= [hodnota]
- c) jmeno\_promenne<[hodnota]
- d) \$jmeno\_promenne:= [hodnota]

Uživatel zavedené proměnné se v shellu dědí do potomků

- a) vždy
- b) nikdy
- c) jen proměnné označené příkazem export
- d) jen proměnné označené příkazem import
- e) jen proměnné označené příkazem child

Jaké je nesprávné použití obsahu proměnné ADRESAR?

- a) echo \$ADRESAR/NEW
- b) echo \${ADRESAR}NEW
- c) echo \$ADRESAR NEW
- d) echo \$ADRESARNEW

Co provádí příkaz set spuštěný bez voleb a argumentů?

- a) vyprázdní všechny proměnné
- b) smaže všechny proměnné
- c) vypíše všechny proměnné
- d) exportuje všechny proměnné

Ve kterém způsobu závorkování se provede příkaz ve vnořeném shellu?

- a) (prikaz)
- b) { prikaz; }
- c) &lt;prikaz;&gt;

Příkaz **cd `echo \$OLDPWD`** se chová stejně jako:

- a) pwd
- b) cd ~
- c) cd -
- d) cd ..

Který z příkazů má správnou syntaxi?

- a) { echo ahoj;}
- b) { echo ahoj }
- c) {echo ahoj};
- d) {echo ahoj}
- e) {echo ahoj }

Máme nastavený pracovní adresář /tmp/data. Které z použití příkazu **pwd** vypíše jiný pracovní adresář než /tmp/data?

- a) cd \$PWD;pwd
- b) { cd \$PWD;};pwd
- c) { cd \$HOME;};cd -;pwd
- d) { cd \$HOME;};pwd
- e) (cd \$HOME);pwd

Co je uloženo v proměnné PWD

- a) pracovní adresář
- b) heslo, zašifrované jednosměrnou šifrou
- c) domovský adresář
- d) hesla všech uživatelů (zašifrovaná jednosměrnou šifrou)

Mezi složené příkazy nepatří

- a) for
- b) case
- c) while
- d) until
- e) find

Co vykonává příkaz true?

- a) vrací hodnotu true
- b) vypisuje řetězec true dokud není ukončen
- c) vrací návratový kód 0
- d) vrací návratový kód 1
- e) není to příkaz, je to logická hodnota

[ je

- a) příkaz
- b) otevírací závorka pro zápis podmínky
- c) otevírací závorka pro zápis osmičkového čísla
- d) symbol přesměrování
- e) speciální znak

Který z příkazů nesmaže všechny soubory z běžného adresáře?

- a) for S in \*; do rm \$\$; done
- b) rm `ls`
- c) rm `echo \*`
- d) for S in `ls`; do rm \$\$; done
- e) všechny ostatní příkazy soubory smažou

Kterým příkazem pošleme správně celý text e-mailem na zadanou adresu?

- a) echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík>mail adresa@nekde
- b) (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)>mail adresa@nekde
- c) (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)|mail adresa@nekde
- d) echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík|mail adresa@nekde

Jaký příkaz můžeme použít, chceme-li na standardní výstup předat na řádku vždy jméno souboru, mezeru a znovu jméno souboru s příponou .o pro všechny soubory/položky běžného adresáře?

- a) for JM in \$PWD; do echo \$JM \${JM}.o; done
- b) for \$JM in \*; do echo \$JM \$JM.o; done
- c) for JM in \*; do echo \$JM \$JM.o; done
- d) for JM in \*; do echo JM JM.o; done

Jaká je korektní syntaxe zápisu vyhodnocení podmínky?

- a) if `ls soubor`; then rm soubor; fi
- b) if `ls soubor`; then rm soubor; fi
- c) if ls soubor; then rm soubor; fi
- d) if [ ls soubor ]; then rm soubor; fi

Kterým příkazem **ne** zjistím, zda položka 'adresar' je adresář?

- a) if test -d adresar; then echo ok; fi
- b) if [ -d adresar ]; then echo ok; fi
- c) if ls -ld adresar|grep -q '^d'; then echo ok; fi
- d) if pwd adresar; then echo ok; fi

Který příkaz je syntakticky špatně?

- a) if [ \$# -ge 3 ] && [ \$# -lt 6 ]; then echo ok; fi
- b) if [ \$# -ge 3 && \$# -lt 6 ]; then echo ok; fi
- c) if [ \$# -ge 3 -a \$# -lt 6 ]; then echo ok; fi

Při kterém volání skriptu

**#!/bin/bash<br />if [ \$# = 4 -a \$1 != \$2 ]; then echo ok; fi**

se předá na standardní výstup 'ok'?

- a) **./skript 'a' `echo a` a A**
- b) **./skript 1 2 " 3 4**
- c) **./skript 1 2 " 4**
- d) **./skript a \a b \b**

Který soubor nevypíše příkaz **ls x?[a-c]\***

- a) žádný soubor, který by měl jméno delší než 4 znaky
- b) soubor x1.c i když se v adresáři nachází
- c) soubor xabc i když se v adresáři nachází
- d) soubor x?abc\* i když se v adresáři nachází

Jaká je správná syntaxe dotazu na existenci souboru?

- a) if [ -f soubor ] then echo ano fi
- b) if [ -f soubor ] then; echo ano fi;
- c) if [ -f soubor ]; then echo ano; fi
- d) if [ -f soubor; ]; then echo; ano; fi
- e) if [ -f soubor; ]; then echo ano; fi;

Který příkaz je syntakticky správně a současně obsahuje co nejméně mezer?

- a) [-d adresar]&&echo ano
- b) [-d adresar]&&echo ano
- c) [-d adresar ]&&echo ano
- d) [-d adresar ] &&echo ano
- e) [-d adresar ] && echo ano

Pokud proměnná A obsahuje číslo větší než 5, který příkaz podmínku **A > 5** korektně vyhodnotí a předá na standardní výstup 'ano'?

- a) if [ \$A > 5 ] then echo ano; fi;
- b) if [ \$A > 5 ]; then echo ano; fi
- c) if [ \$A -gt 5 ] then echo ano; fi;
- d) if [ \$A -gt 5 ]; then echo ano; fi

Kterým/i znakem/znaky začínají v shellu komentáře?

- a) #
- b) //
- c) !
- d) %
- e) /\*

Kterým příkazem spustím skript v běžném shellu?

- a) /skript
- b) .skript
- c) . skript
- d) ! skript
- e) žádná jiná odpověď není správná

Chci-li spustit skript příkazem **./skript**, musím mít na soubor se skriptem právo minimálně

- a) spuštění
- b) čtení a spuštění
- c) čtení, zápis a spuštění
- d) čtení
- e) zápis a spuštění

Mějme skript

**#!/bin/sh**

**echo \$4\$1\$3\$2**

Co předá na standardní výstup příkaz **./skript a "" b f**

- a) f a b
- b) fab
- c) af b
- d) afb
- e) b a f

Co předá na standardní výstup příkaz "echo \$2", jestliže byl předtím proveden příkaz "set -- a b c"?

- a) --
- b) a
- c) b
- d) c

Co předá na standardní výstup příkaz (**exit 1; exit 0**); **echo \$?**

- a) nic, shell se ukončí
- b) 0
- c) 1
- d) ?

Pracovní soubor v adresáři /tmp s unikátním jménem (žádný jiný aktivní proces spuštěný z téhož skriptu nepoužije stejné jméno) vytvořím příkazem

- a) touch /tmp/pomocny
- b) touch /tmp/pomocny.\$?
- c) touch /tmp/pomocny.\$\$
- d) touch /tmp/pomocny.%
- e) touch /tmp/pomocny.\$!

Příkaz **ls ~** předá na standardní výstup položky z

- a) běžného adresář
- b) domovského adresáře
- c) kořenového adresáře
- d) adresáře /tmp
- e) adresáře /bin

V běžném adresáři máte položky

**a**

**aa**

**amanda**

**am da (mezera ve jméně)**

**AMANDA**

Která jména předá na standardní výstup příkaz

**ls a\*a**

- a) aa amanda am da
- b) aa amanda
- c) amanda
- d) aa amanda am da AMANDA
- e) a aa amanda am da AMANDA

V běžném adresáři máte položky

**a**

**aa**

**amanda**

**am da (mezera ve jméně)**

**AMANDA**

Která jména předá na standardní výstup příkaz

**echo [a-m][!0-9]**

- a) a
- b) aa
- c) amanda am da
- d) amanda am da AMANDA
- e) [a-m][!0-9]

Skript, který se spustí poté, co se přihlásím do systému, je uložen v souboru

- a) /home/.profile
- b) /home/profile
- c) ~/.profile
- d) ~/profile
- e) /etc/login

Mám v běžném adresáři soubory .kshrc, dopis.txt, a.out, prog.c. Co předá na standardní výstup příkaz 'echo \*' ?

- a) \*
- b) .kshrc dopis.txt a.out prog.c
- c) dopis.txt a.out prog.c .kshrc
- d) .kshrc
- e) a.out dopis.txt prog.c

Mám v běžném adresáři jména souborů a, a1, aa1, 1a

Která jména souborů předá na standardní výstup příkaz 'ls a[a1]\*'?

- a) a a1 aa1
- b) aa1 a1
- c) 1a a a1 aa1
- d) aa1 a1 1a

Který příkaz předá na standardní výstup číslo 4?

- a) echo 2+2
- b) echo A:=2+2;
- c) echo `(2+2)`
- d) echo \${2+2}
- e) echo \$2+\$2

Příkaz A=5;B=2;echo C=\$A+\$B předá na standardní výstup

- a) 7
- b) C=7
- c) C=5+2
- d) C=A+B
- e) nic

Který skript se spouští vždy při spuštění nového shellu bash?

- a) ~/.profile
- b) ~/.bashrc
- c) /bash/bashrc
- d) /bin/bash/profile

Jakým příkazem předám na standardní výstup hodnotu prvního pozičního parametru?

- a) echo \$1
- b) cat \$1
- c) \$1
- d) set \$1
- e) cat '\$1'

Vytvořte adresář zadaný prvním pozičním parametrem.

Který příkaz jej nevytvoří?

- a) if [ ! -d \$1 ]; then mkdir \$1; fi
- b) test -d \$1 || mkdir \$1
- c) [ ! -d \$1 ] && mkdir \$1
- d) if test ! -d \$1; then mkdir \$1; fi
- e) [ -d \$1 ] && mkdir \$1

Jakým příkazem spustíte skript v běžném adresáři a jako první poziční parametr mu předáte řetězec **ahoj**?

- a) ./skript 1=ahoj
- b) ./skript \$1=ahoj
- c) \$1=ahoj; ./skript
- d) ./skript;set -- ahoj
- e) ./skript ahoj

Kterým příkazem smažu všechny soubory zadané pozičními parametry (pozičních parametrů může být libovolné množství)?

- a) rm \$\*
- b) rm ` \$\*`
- c) rm \*
- d) rm \$1 \$2 \$3 \$4 ...
- e) rm `\*`

Jak NEspustíme skript, abychom mu jako poziční parametry předali jména položek běžného adresáře začínající písmenem A?

- a) ./skript `echo A\*`
- b) ./skript 'ls A\*'
- c) ./skript A\*

Příkaz **echo \$\$** předá na standardní výstup

- a) jméno spuštěného skriptu
- b) číslo procesu shellu
- c) počet pozičních parametrů předaných skriptu
- d) všechny poziční parametry předané skriptu

Jaká minimální oprávnění jsou nutná pro spuštění shellového skriptu příkazem **bash skript**?

- a) právo spuštění/provedení
- b) právo čtení
- c) právo spuštění/provedení a právo čtení
- d) právo spuštění/provedení a právo zápisu
- e) právo spuštění/provedení a právo čtení a právo zápisu

Jaká je správná syntaxe příkazu pro porovnání numerického obsahu dvou proměnných?

- a) [ \$A > \$B ]
- b) [ \$A -gt \$B ]
- c) [ \$A>\$B ]
- d) [ \$A-gt\$B ]

Příkaz **touch knihovna; test -d knihovna** na jinak prázdném adresáři vrátí návratový kód

- a) prázdný
- b) 0
- c) 1

Příkazem **PS1=ferda** změníme

- a) implicitní podpis v e-mailu
- b) nastavení běžného adresáře
- c) výzvu na terminálu uživatele
- d) jméno mailboxu
- e) primární aktivní proces

Příkazovou substituci uzavíráme do

- a) '...'
- b) "..."
- c) `...`
- d) <...>

Operátorem **2>**

- a) přesměrujeme standardní výstup
- b) přesměrujeme standardní chybový výstup
- c) přesměrujeme standardní vstup
- d) zavřeme standardní výstup
- e) zrušíme chybový výstup

Kterým příkazem vypíšeme na standardní výstup jména všech adresářových položek, které kdekoli ve svém jméně mají písmeno A?

- a) echo \*.A.\*
- b) ls '\*.A\*'
- c) echo \*A\*
- d) ls ".A.\*"
- e) ls \*A.\*

Jakým příkazem spojíme za sebe obsah souborů **a** a **b** a výsledek uložíme do souboru **c**?

- a) `cat < a < b > c`
- b) `cat a b | c`
- c) `cat a b > c`
- d) `cat < a < b | c`
- e) `a b | cat > c`

Vytvořte adresář **Mail** v domovském adresáři uživatele, pokud tam již není. Jaký příkaz toto NEprovede?

- a) `[ -d ~/Mail ] || mkdir ~/Mail`
- b) `(cd; test -d Mail || mkdir Mail)`
- c) `if [ ! -d $HOME/Mail ]; then mkdir $HOME/Mail; fi`
- d) `(cd ~; mkdir Mail 2>/dev/null)`
- e) `test -d ~/Mail && mkdir Mail 2>/dev/null`

Jakým příkazem zkopírujete obsah souboru **a** do souborů **b**, **c**, **d** současně?

- a) `cat a > b > c > d`
- b) `cp a b c d`
- c) `tee < a > b > c > d`
- d) `cat a | tee b c > d`
- e) `tee a | cat b c d`

Jakým příkazem přidáte za konec souboru **x** řádek obsahující **xxx-konec-xxx**?

- a) `echo xxx-konec-xxx >> x`
- b) `cat x < xxx-konec-xxx`
- c) `cat x|echo xxx-konec-xxx > x`
- d) `cat xxx-konec-xxx << x`

Jakým příkazem předáte na standardní výstup všechna jména souborů z běžného adresáře, jejichž přípona jména souboru končí na **.txt**? Všechna předaná jména souborů musí být nutně na jednom řádku oddělená vzájemně bílým místem.

- a) `for S in *.txt; do echo $S; done`
- b) `cat *.txt`
- c) `echo *.txt`
- d) `ls -l *.txt`