UVOD

Operační systém UNIX vznikl okolo roku
1) 1950
2) 1960
3) 1970
4) 1980
5) 1990
// 3 ok

První verze UNIXu byla naprogramována pro počítač
1) IBM PC
2) PDP
3) IBM 370
4) VAX
5) Sun
// 2 ok

První verze UNIXu byla vytvořena v
1) Bell Laboratories
2) University of Berkeley
3) Microsoftu
4) Novellu
5) USL (UNIX System Laboratories)
// 1 ok

Co vzniklo dříve? MS-DOS (předchůdce MS-Windows) nebo UNIX?
1) MS-DOS
2) UNIX
3) Obojí je zhruba stejně staré.
// 2 ok

Kdo jsou autoři prvních verzí UNIXu?
1) Steeve Jobs, Tim O'Reilly, Andrew Tanenbaum
2) Larry Ellison, Scott McNealy, Michael Tiemann
3) Brian Kernighan, Dennis Ritchie, Ken Thompson
4) Eric Raymond, Richard Stallman, Linus Torvalds
// 3 ok

Operační systém Linux vzniká od roku
1) 1951
2) 1961
3) 1971
4) 1981
5) 1991
// 5 ok

Záznamy o uživatelích, kteří mají právo přistupovat k systému, jsou uloženy v souboru:
1) /etc/group
2) /etc/users
3) /etc/passwd
4) /etc/shadow
5) /etc/motd
// 3 ok

Hesla uživatelů v otevřeném tvaru jsou uložena v uživatelům nepřístupném souboru:
1) /etc/passwd
2) /etc/passwords
3) /etc/shadow
4) nejsou uložena
// 4 ok

Soubor /etc/passwd v aktuálních verzích systému
1) obsahuje zašifrovaná hesla uživatelů
2) neobsahuje hesla uživatelů
3) nikdy, ani v minulosti neobsahoval hesla uživatelů
4) obsahuje nezašifrovaná hesla uživatelů
// 2 ok

Účet uživatele neobsahuje
1) primární skupinu
2) uživatelské jméno
3) UID uživatele
4) UČO uživatele
5) žádná z ostatních odpovědí není správná
// 4 ok

Hesla uživatelů v zašifrovaném tvaru jsou uložena v uživatelům nepřístupném souboru:
1) /etc/passwd
2) /etc/shadow
3) /etc/passwords
4) nejsou uložena
// 2 ok

Uživatel zapomněl svoje heslo. Jak jej nejsnáze získá?
1) uživatel použije příkaz decryptpas
2) uživatel požádá superuživatele o provedení příkazu decryptpas
3) uživatel požádá superuživatele o získání svého hesla ze souboru /etc/shadow
4) žádná jiná odpověď není správná
// 4 ok

Uživatelské jméno se nazývá:
1) login
2) loggin
3) user name
4) uid
5) user
// 1 ok

Hodnota UID u uživatele root je
1) A
2) root
3) 0
4) 1
5) 99999
// 3 ok

Uživatel je členem
1) nemusí být členem žádné skupiny
2) je členem alespoň jedné skupiny
3) je členem alespoň dvou skupin
// 2 ok

Seznam skupin uživatelů v systému je uložen v souboru
1) /etc/team
2) /etc/groups
3) /etc/group

4) /etc/teams
5) /etc/cluster
// 3 ok

Soubor /etc/passwd musí být
1) čitelný pouze superuživateli
2) nesmí být čitelný nikomu
3) čitelný pro všechny
4) čitelný pouze skupině root
5) žádná jiná odpověď není správná
// 3 ok

Soubor /etc/shadow musí být
1) čitelný pouze superuživateli
2) nesmí být čitelný nikomu
3) čitelný pro všechny
4) žádná jiná odpověď není správná
// 1 ok

Každé uživatelské jméno v souboru /etc/passwd musí být definováno
1) alespoň jednou
2) právě jednou
3) nemusí být definováno vůbec
4) právě dvakrát
5) alespoň dvakrát
// 2 ok

Každé uživatelské jméno v souboru /etc/group musí být použito
1) alespoň jednou
2) právě jednou
3) nemusí být uvedeno vůbec
4) právě dvakrát
5) alespoň dvakrát
// 3 ok

Interpret příkazů a domácí adresář, který se spustí a nastaví po přihlášení uživatele, je popsán
1) v souboru .profile
2) v účtu uživatele
3) stavem při posledním odhlášení
4) v souboru /etc/userdef
5) v UID záznamu
// 2 ok

Čas v unixu je uložen (např. čas změny obsahu souboru)
1) ve tvaru yyyymmddhhmiss
2) ve tvaru ddmmyyyyhhmiss
3) v počtu hodin od začátku epochy
4) v počtu minut od začátku epochy
5) v počtu sekund od začátku epochy
// 5 ok

Epocha v unixu začala
1) 1. 1. 1970 00:00
2) 31. 12. 1970 24:00
3) 1. 1. 1980 00:00
4) 31. 12. 1980 24:00
// 1 ok

Počet údajů na řádku v souboru /etc/passwd je (některý může být nevýznamný) alespoň
1) 4
2) 5
3) 6
4) 7
5) 8
// 4 ok

Odlišnosti v chování textových terminálů jsou
1) sjednoceny instalací kompatibilního driveru
2) transformovány na jednotné volání popisem v databázi /lib/terminfo/*/*
3) omezeny na kompatibilní typy terminálů v compatibility seznamu
// 2 ok

Démon je
1) je ovladač spících zařízení s nejnižší prioritou
2) je spící program aktivovaný událostí
3) infiltrovaný interpret příkazů
4) ladící systém jádra operačního systému
// 2 ok

Každý řádek textového souboru v unixu končí
1) znakem CR
2) znakem LF
3) znaky CR,LF
4) znaky LF,CR
// 2 ok

Každý řádek textového souboru v unixu končí
1) znakem Carriage Return
2) znakem Long Field
3) znaky CR,LF
4) znaky LF,CR
5) žádná z ostatních odpovědí není správná
// 5 ok

Každý řádek textového souboru v MS-DOSu/Windows končí
1) znakem CR
2) znakem LF
3) znaky CR,LF
4) znaky LF,CR
// 3 ok

Při přenosu textového souboru z MS-DOSu/Windows do unixu
1) přidáváme řídící znak CR za každý řádek
2) přidáváme řídící znak LF za každý řádek
3) odebíráme řídící znak CR za každým řádkem
4) odebíráme řídící znak LF za každým řádkem
// 3 ok

Při přenosu textového souboru z unixu do MS-DOSu/Windows
1) přidáváme řídící znak CR za každý řádek
2) přidáváme řídící znak LF za každý řádek
3) odebíráme řídící znak CR za každým řádkem
4) odebíráme řídící znak LF za každým řádkem
// 1 ok

Primární prompt běžného uživatele unixu je znak
1) #
2) $
3) &
4) *

5) %
// 2 ok

Primární prompt superuživatele je znak
1) #
2) $
3) &
4) *
5) %
// 1 ok

Primární prompt si uživatel nastavuje/mění v proměnné prostředí
1) PROMPT
2) PRMPT
3) PRM1
4) PS1
5) PS
// 4 ok

Příkazem stty -a
1) se uživatel odhlásí ze sezení u terminálu
2) se uživateli obnoví výchozí nastavení terminálu
3) se uživateli sdělí nastavení terminálu
4) se uživateli zpřístupní všechny terminály
5) se uživateli přepne terminál do ascii režimu
// 3 ok

Konvence pro prezentaci (výpis) znaku control-c je
1) CTRL-C
2) CC
3) ^C
4) C-C
5) #C
// 3 ok

Výpis (prezentace) \012 znamená
1) zápis hodnoty -12
2) znak s ordinální hodnotou 12 dekadicky
3) znak s ordinální hodnotou 12 oktalově
4) znak s ordinální hodnotou 12 hexadecimálně
5) textový řetězec 12
// 3 ok

Speciální znak intr (typicky control-c)
1) přeruší činnost operačního systému
2) přeruší běh procesu na popředí
3) přeruší výpis na terminál na popředí
4) přeruší výpis na terminál na pozadí
// 2 ok

Proces běžící na popředí násilně ukončím (např. pokud cyklí v nekonečné smyčce) stiskem speciálního znaku
1) intr
2) stop
3) kill
4) eof
// 1 ok

Speciální znak eof (typicky control-d)
1) ukončí stav X-off
2) přeruší běh procesu
3) ukončí zpracování fail stavu
4) vloží příznak konce souboru
// 4 ok

Speciální znaky start a stop (typicky control-q a control-s)
1) realizují protokol X-ON/X-OFF
2) spustí a ukončí proces
3) přihlásí a odhlásí uživatele
4) spustí a ukončí komunikaci operačního systému
// 1 ok

Při provádění příkazu cp /dev/tty /dev/null nezískáte prompt stisknutím speciálního znaku
1) stop
2) eof
3) intr
4) susp
// 1 ok

Co provede příkaz cp /etc/passwd /dev/tty
1) zkopíruje obsah běžného souboru /etc/passwd do běžného souboru /dev/tty
2) zkopíruje obsah běžného souboru /etc/passwd na první tiskárnu systému
3) zkopíruje obsah běžného souboru /etc/passwd na terminál uživatele
4) čte z klávesnice terminálu a zapisuje do souboru /etc/passwd
// 3 ok

Přihlašovací (login) shell nelze za žádných okolností ukončit
1) speciálním znakem eof
2) speciálním znakem stop
3) speciálním znakem intr
4) příkazem exit
5) rozpadnutím spojení
// 2 ok

Která služba používá šifrovanou komunikaci mezi klientem a serverem?
1) telnet
2) rsh
3) ssh
4) ftp
// 3 ok

Jaký počet kořenových adresářů je uživateli dostupný?
1) tolik, kolik je v systému připojených diskových zařízení
2) tolik, kolik je v systému připojených diskových zařízení, plus 1
3) právě jeden
4) žádný
// 3 ok

Při úspěšném přihlášení uživatele do systému se neprovede
1) vypíší se systémové zprávy, jsou-li nějaké
2) vypíše se sdělení správce systému, je-li nějaké
3) spustí se uživatelem vybraný shell dle obsahu /etc/passwd
4) uživatel se přihlásí do skupiny dle obsahu /etc/group
5) vypíše se prompt shellu
// 4 ok

Při zadávání příkazů shellu neplatí, že

1) lze příkazy zadávat "do zásoby"
2) lze u některých shellů využít paměti starých (historii) příkazů
3) příkazy lze zadávat bez ohledu na velikost písmen
4) po stisknutí klávesy Enter již nelze opravovat obsah odeslaného řádku
5) řádek lze upravovat tak dlouho, dokud není stisknuta klávesa Enter
// 3 ok

Při zadávání příkazů shellu klávesou Enter ještě neodešleme příkaz shellu ke zpracování (použijeme pokračovací řádek), když
1) klávesu Enter stiskneme na konci řádku
2) klávesu Enter stiskneme rychle dvakrát po sobě (tzv. doubleclick)
3) před stiskem klávesy Enter vložíme obrácené lomítko
4) po stisku klávesy Enter vložíme znak větší-než
// 3 ok

Sekundární prompt se používá
1) ve vnořeném shellu
2) na pokračovacím řádku
3) při čtení příkazů ze skriptu
4) na řádku zrušeném speciálním znakem kill
// 2 ok

Sekundární prompt lze přenastavit změnou proměnné prostředí PS2
a implicitně obsahuje znak
1) $
2) >
3) #
4) &
5) *
// 2 ok

Historii mnou zadaných příkazů shellu bash mi vypíše příkaz
1) man history
2) cat history
3) cat ~/.history
4) cat ~/.bash_history
5) žádná z ostatních odpovědí není správně
// 4 ok

Historie zadaných příkazů shellu bash se ukládá do souboru s historií příkazů, když
1) se uživatel přihlašuje
2) uživatel zadává jakýkoli příkaz
3) se uživatel odhlašuje
4) uživatel zadá příkaz history
// 3 ok

Operační systém Linux
1) vznikl předáním ochranné známky UNIX společnosti GNU
2) je předchůdcem unixu
3) vznikl nezávisle na unixu
4) je následovníkem unixu
// 3 ok

Uživatel ke v/v zařízením v unixu přistupuje prostřednictvím
1) instalovaných driverů
2) ikon zařízení
3) speciálních souborů
4) přerušení
// 3 ok

UNIX byl při svém vzniku koncipován jako systém
1) multiprogramový, multiuživatelský, síťový
2) multiprogramový, multiuživatelský, s terminálovým přístupem
3) multiprogramový, síťový, s terminálovým přístupem
4) multiuživatelský, síťový, s terminálovým přístupem
// 2 ok

Ukončí-li se shell, který byl spuštěn po přihlášení uživatele
1) je uživatel odhlášen ze systému
2) automaticky se spustí další shell, není-li zadán příkaz exit
3) tento shell uživatel nemůže ukončit, protože je systémový
4) systém se uživatele zeptá, zda chce spustit novou kopii shellu
// 1 ok

Bezprostředně po ukončení právě prováděného příkazu shellu, jsou-li zadány nějaké příkazy "do zásoby"
1) se příkazy v zásobě ihned provedou bez potvrzení uživatele
2) musí uživatel potvrdit provedení každého příkazu v zásobě zvlášť
3) musí uživatel potvrdit celou skupinu příkazů v zásobě, jsou provedeny bud všechny nebo ani jeden
4) nelze zadávat příkazy do zásoby bez povolení superuživatele
// 1 ok

Pro šifrované zpřístupnění terminálu vzdáleného počítače použijeme
1) telnet
2) ssh
3) rsh
4) ftp
5) scp
// 2 ok

Které tři z vyjmenovaných aplikací patří mezi unixové shelly?
1) bash, ssh, ksh
2) ssh, telnet, putty
3) sh, bash, ksh
4) bash, sh-bash, ksh-bash
// 3 ok

Proces běžící na popředí pozastavím speciálním znakem
1) intr
2) eof
3) susp
4) supr
5) su
// 3 ok

Nápovědu k příkazu rm získám pomocí
1) rm -f
2) rmman
3) rm help
4) man rm
// 4 ok

Proces je v unixu jednoznačně identifikován

1) uživatelem, který spustil
2) číslem přiřazeným při spuštění
3) terminálem, ze kterého byl spuštěn
4) časem spuštění
// 2 ok

Speciální znak susp
1) přepne počítač do úsporného režimu
2) pozastaví proces
3) ukončí proces
4) zamkne relaci uživatele
// 2 ok

Příkaz ps
1) vypíše přihlášené uživatele
2) vypíše spuštěné procesy
3) vypíše programy v systému souborů
// 2 ok

Příkaz kill -9 1000
1) násilně ukončí program s názvem 1000, mám-li na to právo
2) násilně ukončí program s názvem 1000
3) násilně ukončí proces s číslem 1000, mám-li na to právo
4) násilně ukončí proces s číslem 1000
5) je chybný příkaz
// 3 ok

Uživatel root
1) smí číst všechny soubory, nesmí je však modifikovat
2) smí číst všechny soubory a modifikovat jen ty, které jsou ve skupině root
3) uživateli root se přístupová práva nekontrolují
// 3 ok

Uživatelů root smí být v systému zavedeno
1) žádný
2) jeden
3) dva
4) tolik, kolik jich je zavedeno v souboru /etc/passwd
// 2 ok

Příkaz man cat
1) vypíše obsah souboru cat na standardní výstup
2) vypíše obsah souboru man na standardní výstup
3) vypíše manuálovou stránku příkazu cat na standardní výstup
4) vypíše manuálovou stránku příkazu man na standardní výstup
5) je chybný příkaz
// 3 ok

Příkaz man 5 passwd
1) může vypsat manuálovou stránku příkazu passwd
2) může vypsat manuálovou stránku souboru /etc/passwd
3) může vypsat manuálovou stránku příkazu login
4) je chybný příkaz
// 2 ok

Kterým příkazem nelze získat obsah (předpokládejme textového) souboru
1) cat
2) more
3) less
4) pwd
5) všemi uvedenými
// 4 ok

Adresář /etc je určen
1) pro dočasné soubory jako pomocný
2) pro různé soubory, které nebylo možné dát do jiných adresářů
3) pro umístění zpravidla konfiguračních souborů
4) pro umístění zpravidla speciálních souborů
// 3 ok

Externí příkazy shellu jsou zpravidla umístěny v adresáři
1) /usr/local/bin
2) /bin
3) /etc
4) /usr
5) /usr/lib
// 2 ok

Pro umísťování adresářů se soubory, jejichž obsah se často mění (např. log soubory se záznamy o provedených operacích), je určen adresář
1) /bin
2) /lib
3) /usr
4) /var
5) /etc
// 4 ok

FILE SYSTEM

Za základní systém souborů považujeme, ten
1) který je nejrozšířenější z podporovaných
2) který je na hlavním disku systému
3) který je na disku C:
4) který obsahuje kořenový adresář systému
// 4 ok

Mezi základní systémy souborů patří
1) iso8859-2, sysfs
2) nfs, iso9660
3) ext3, xfs
4) vfat, tmpfs
// 3 ok

Mezi typické doplňující systémy souborů patří
1) vfat, iso9660, ntfs
2) swap, ext3, tmpfs
3) ntfs, iso8859, xfs
// 1 ok

V souboru /etc/fstab je
1) výstup kontroly systému souborů příkazem fsck
2) seznam vadných bloků na zařízeních
3) připojení systémů souborů k zařízením
4) seznam uživateli dostupných systémů souborů
5) takový soubor neexistuje
// 3 ok

Při neřízeném vypnutí stroje (např. při výpadku napájení) může dojít ke ztrátě dat zapisovaných do systému souborů
1) plánovaných zapsat po výpadku
2) během výpadku a plánovaných zapsat po výpadku
3) před výpadkem, během výpadku a plánovaných zapsat po výpadku
// 3 ok

Operace spojená s poznačováním obsahu vyrovnávacích pamětí se nazývá typicky
1) set
2) save
3) flush
4) write
// 3 ok

Jméno souboru či adresáře smí být dlouhé nejvýše
1) 8+3 znaků
2) 16+3 znaků
3) 128 znaků
4) 255 znaků
// 4 ok

Maximální délka souboru v unixu je
1) 255 znaků
2) 256 znaků
3) 128 znaků
4) je omezena vlastnostmi konkrétního systému souborů
// 4 ok

Soubory, jejichž jméno začíná tečkou
1) jsou přístupné pouze superuživateli
2) se nezahrnují do *-expanze
3) se nepoužívají
4) takové jméno souboru není možné
// 2 ok

Příkaz ls -la nevypíše
1) čas změny souboru
2) číslo i-uzlu, ve kterém je uložen příslušný soubor
3) velikost souboru
4) soubory, jejichž jméno začíná tečkou
5) adresářové položky . a ..
// 2 ok

Který příkaz vypíše počet odkazů na jednotlivé soubory?
1) ls
2) ls -l
3) ls -a
4) ls -i
// 2 ok

Relativní cesta k souboru nebo k adresáři začíná vždy
1) lomítkem
2) běžným adresářem
3) domovským adresářem
4) znakem ~
// 2 ok

Rozdíl mezi absolutní a relativní cestou k souboru či adresáři je
1) absolutní cesta začíná vždy domovským adresářem
2) absolutní cesta začíná vždy běžným adresářem
3) absolutní cesta začíná vždy lomítkem
// 3 ok

K oddělování adresářů a jména souboru v zápisu cesty se používá znak
1) \
2) /
3) |
// 2 ok

V hierarchii adresářů stojí nejvýše adresář
1) /
2) C:/
3) C:/, D:/, E:/, ...
4) root
5) /root
// 1 ok

Běžný adresář jako domovský si uživatel nastaví
1) příkazem cd
2) editací souboru /etc/passwd
3) příkazem pwd
4) nenastaví
// 4 ok

Kterým příkazem se změní běžný adresář vždy na můj domovský adresář?
1) cd ~
2) cd -
3) cd .
4) cd /
// 1 ok

Co se stane po provedení příkazu "cd -"?
Bude nastaven
1) předchozí pracovní adresář
2) domovský adresář aktuálního uživatele
3) domovský adresář superuživatele
4) speciální adresář definovaný v /etc/passwd
// 1 ok

Prázdný adresář je adresář
1) je adresář s nulovou velikostí
2) je adresář s prazdnou tabulkou adresáře
3) obsahující dvě konkrétní položky
// 3 ok

Cestu k běžnému adresáři předá na standardní výstup příkaz
1) cd
2) pwd
3) pcd
4) wd
5) grep $LOGNAME /etc/passwd|cut -d: -f6
// 2 ok

Jestliže v adresáři /home/user zadate příkaz cd ./.. , potom váš běžný adresář bude
1) /home
2) /
3) nelze zadat více argumentů, vypíše se chyba
4) /home/user
5) domovský adresář
// 1 ok

Po zadání příkazu cd /tmp/data v adresáři /home/user se změní běžný adresář na
1) /home/user/tmp/data
2) /home/tmp/data
3) /tmp/data
4) /tmp/data/home/user
// 3 ok

Pracovní adresář je nastaven na /home/user/data. Jaký bude pracovní adresář po provedení příkazu cd ./../../.
1) /
2) /home
3) /home/user
4) /home/user/data
// 2 ok

Vytvoření a zrušení adresáře provedou příkazy
1) md a rd
2) mdir a rdir
3) mkdir a rmdir
4) make a remove
// 3 ok

Nemám-li povoleno v proměnné PATH prohledávání běžného adresáře, spustím proveditelný soubor muj příkazem
1) run muj
2) ../muj
3) ./muj
4) muj
// 3 ok

Adresář v systému souborů je uložen v souboru
1) adresář není uložen v souboru
2) obsahujícím čísla i-uzlu
3) obsahujícím jména souborů a čísla i-uzlu
4) obsahujícím cestu, jména souborů a čísla i-uzlu
5) obsahujícím identifikaci majitele, cestu, jména souborů a čísla i-uzlu
// 3 ok

Kořenový adresář má číslo i-uzlu
1) 0
2) 1
3) 2
4) 3
5) -1
// 3 ok

Kořenový adresář je v UNIXu označen
1) /root
2) \root
3) \
4) /
5) C:\
// 4 ok

Položka '.' má v adresáři přiřazené číslo i-uzlu
1) 0
2) 1
3) 2
4) shodné s číslem i-uzlu tohoto adresář
5) shodné s číslem i-uzlu rodičovského adresáře
// 4 ok

Pokud je ve výpisu obsahu adresáře u adresáře "." uvedeno číslo i-uzlu 2 a u adresáře ".." 0, potom se jedná o
1) běžný (pracovní) adresář
2) nesmysl
3) adresář druhé úrovně např. /home
4) adresář třetí úrovně např. /home/user
// 2 ok

Příkazem rmdir se typicky maže adresář obsahující
1) 0 položek
2) 1 položku
3) 2 položky
4) 3 položky
// 3 ok

i-uzel
1) obsahuje adresu počítače v síti
2) obsahuje informace o směrování paketů v uzlech sítě
3) obsahuje informaci o jednom souboru či adresáři systému souborů
4) obsahuje informaci o jednom uzlu n-uzlového clusteru
// 3 ok

Zdrojový soubor zruší příkaz
1) mv
2) cp
3) cat
4) ls
// 1 ok

Soubory rušíme příkazem
1) mv
2) cp
3) rm
4) cat
5) del
// 3 ok

Systém vždy fyzicky odstraní soubor z tabulky i-uzlů (tzn. soubor se zruší), když
1) provedu příkaz rm
2) počet odkazů na počítadle v i-uzlu klesne na 0
3) se uživatel odhlásí, poté co provedl příkaz rm
4) se vysype koš
5) fyzicky zruším i-uzel příkazem rmnode
1) příkazem rm odstraníte jen jeden z odkazů, další ještě mohou existovat
// 2 ok

Co se stane po provedení příkazu "rm -rf *"?
Bude smazán obsah pracovního adresáře
1) vč. jmen začínajících tečkou a vč. podadresářů a bez všech varování
2) vč. jmen začínajících tečkou, ale bez podadresářů
3) bez jmen začínajících tečkou, vč. podadresářů a bez všech varování
4) bez jmen začínajících tečkou, bez podadresářů a bez všech varování
5) vč. jmen začínajících tečkou a vč. podadresářů s varováními
// 3 ok

i-uzel obsahuje

1) všechny informace o souboru vyjma dat souboru
2) všechny informace o souboru vyjma cesty k souboru a dat
3) všechny informace o souboru vyjma jména souboru, cesty k souboru a dat
4) všechny informace o souboru vyjma identifikace majitele, jména souboru, cesty k souboru a dat
5) všechny informace o tomto uzlu v síti
// 3 ok

Kořenový adresář má pro položky '.' a '..'
1) čísla i-uzlu o jedničku větší
2) čísla i-uzlu o jedničku menší
3) čísla i-uzlu stejná
4) žádná z ostatních odpovědí není správná
// 3 ok

Pokud příkazem mv přesouváte soubory v rámci jednoho systému souborů, pak se číslo i-uzlu přesouvaného souboru
1) změní
2) nezmění
3) zruší
// 2 ok

Co provede příkaz ls -a (označte nejsprávnější odpověď):
1) vypíše seznam všech podadresářů
2) vypíše seznam všech souborů
3) vypíše seznam všech podadresářů a souborů a položek se jménem začínajícím tečkou
4) vypíše seznam všech položek se jménem začínajícím tečkou
5) provede totéž jako příkaz ls bez parametru
// 3 ok

Kterým příkazem lze smazat neprázdný adresář?
1) rmdir
2) rm -r
3) rm -a
4) rm -f
5) neprázdný adresář nelze smazat
// 2 ok

Který příkaz předá na standardní výstup obsah proměnné PATH?
1) show $PATH
2) echo $PATH
3) echo PATH
4) ls $PATH
// 2 ok

Příkazem pwd
1) změníme pracovní adresář
2) vypíšeme pracovní adresář
3) změníme domovský adresář
4) vypíšeme domovský adresář
// 2 ok

Příkazem rmdir
1) vymažeme prázdný adresář
2) přesuneme prázdný nebo neprázdný adresář do koše
3) přesuneme prázdný adresář do koše
4) vyprázdníme koš
5) vymažeme skrytý adresář
// 1 ok

Maximální počet i-uzlů v systému souborů
1) lze dle potřeby měnit příkazem ls -il
2) lze dle potřeby měnit příkazem mkfs
3) nelze měnit bez nového vytvoření systému souborů
// 3 ok

Počet odkazů na soubor je uložen
1) v adresáři
2) v i-uzlu
3) v souboru
4) není uložen, vždy se při potřebě vypočítává
// 2 ok

Na soubor vytvořený běžným způsobem textovým editorem
1) je nulový počet tvrdých odkazů
2) je jeden tvrdý odkaz
3) jsou dva tvrdé odkazy
// 2 ok

Jakým příkazem vytvoříme tvrdý odkaz na soubor?
1) ln -t soubor odkaz
2) ln -h soubor odkaz
3) ln soubor odkaz
4) ln -s soubor odkaz
// 3 ok

Po provedení posloupnosti příkazů
touch a; ln a b; ln a c
vzniknou následující počty tvrdých odkazů na jednotlivé soubory
1) a 1, b 2, c 3
2) a 3, b 2, c 1
3) a 2, b 2, c 2
4) a 3, b 3, c 3
5) příkaz je chybný
// 4 ok

Kolik tvrdých odkazů ukazuje na soubor v následujícím výpisu příkazu ls -l?
-rw-r r 3 ja oni 2 Mar 1 04:05 cosi
1) 1
2) 2
3) 3
4) 4
5) 5
// 3 ok

Po provedení příkazu ln aa bb přibude v běžném adresáři na výpise příkazu ls -l položka
1) lrw-r--r 1 brandejs staff 6 dub 8 21:50 bb
2) -rw-r--r 1 brandejs staff 6 dub 8 21:50 bb
3) lrw-r--r 2 brandejs staff 6 dub 8 21:50 bb
4) -rw-r--r 2 brandejs staff 6 dub 8 21:50 bb
// 4 ok

Výpis příkazu ls -il:
539520038 -rw-r--r 2 brandejs staff 0 bře 29 13:46 a
539520038 -rw-r--r 2 brandejs staff 0 bře 29 13:46 b
Který z odkazů vznikl dříve?
1) odkaz na soubor a
2) odkaz na soubor b
3) vznikly zároveň
4) nelze určit

5) výpis je nesmyslný
// 4 ok

Výpis příkazu ls -il:
539520038 -rw-r--r 2 brandejs staff 0 bře 29 13:46 a
539520038 -rw-r--r 2 brandejs staff 0 bře 29 13:46 b
Kdo vytvořil tvrdý odkaz 'a' na 'b' nebo 'b' na 'a'?
1) výhradně uživatel brandejs
2) výhradně uživatel staff
3) výhradně člen skupiny staff
4) výhradně člen skupiny brandejs
5) nelze určit
// 5 ok

Který příkaz úspěšně skončí po provedení posloupnosti příkazů
touch a; mkdir b
1) ln a c
2) ln b c
3) žádný z uvedených příkazů
// 1 ok

Uživatel xnovak vytvořil příkazem ln tvrdý odkaz na soubor, který vlastní uživatel xpolak. Kdo bude ve výpise příkazu ls -l uveden jako vlastník odkazu?
1) xnovak
2) xpolak
3) oba
4) nikdo z nich
// 2 ok

Kolik bude tvrdých odkazů na adresář a, provedete-li posloupnost příkazů
mkdir a; mkdir a/b; mkdir a/c; touch a/d
1) 2
2) 3
3) 4
4) 5
5) posloupnost příkazů je nesmyslná
// 3 ok

Kolik bude tvrdých odkazů na adresář a, provedete-li posloupnost příkazů
touch a; touch a/b; touch a/c; touch a/d
1) 2
2) 3
3) 4
4) 5
5) posloupnost příkazů je nesmyslná
// 5 ok

Kolik bude tvrdých odkazů na adresář b, provedete-li posloupnost příkazů
mkdir a; mkdir a/b; mkdir a/c; mkdir a/b/d; mkdir a/c/e
1) 2
2) 3
3) 4
4) 5
5) posloupnost příkazů je nesmyslná
// 2 ok

Kolik bude tvrdých odkazů na adresář b, provedete-li posloupnost příkazů
mkdir a; mkdir a/b; mkdir a/c; mkdir a/b/d; rmdir a/b/d
1) 2
2) 3
3) 4
4) 5
5) posloupnost příkazů je nesmyslná
// 1 ok

Po provedení posloupnosti příkazů
mkdir a; mkdir a/b; ls a; ls -d a
se na standardní výstup předají řádky obsahující
1) a
b
2) b
a
3) a
a
4) b
b
5) posloupnost příkazů je nesmyslná
// 2 ok

Po provedení posloupnosti příkazů
touch a; mkdir b; touch b/a; ls a; ls b
se na standardní výstup předají řádky obsahující
1) a
b
2) b
a
3) a
a
4) b
b
5) posloupnost příkazů je nesmyslná
// 3 ok

Z výpisu příkazu ls -l
-rwxr-xr-x 2 brandejs staff 6 bře 29 23:05 a
drwxr-xr-x 2 brandejs staff 6 bře 29 23:05 b
-rwxr-xr-x 2 brandejs staff 6 bře 29 23:05 c
vyplývá, že
1) a, c jsou soubory, b je speciální soubor
2) a, c jsou soubory, b je adresář
3) a, b jsou adresáře, c je soubor
4) a, c jsou adresáře, b je soubor
5) výpis je nesmyslný
// 2 ok

Tvrdý odkaz se nepovolí udělat na
1) speciální soubor
2) symbolický odkaz
3) adresář
4) soubor
// 3 ok

Symbolický odkaz se nepovolí udělat na
1) adresář
2) soubor v jiném systému souborů
3) kořenový adresář
4) speciální soubor /dev/null
5) i-uzel
// 5 ok

Provedením příkazu ln (bez voleb) se počet odkazů na objekt typicky
1) zvyšuje
2) snižuje
3) nemění
// 1 ok

Provedením příkazu rm se počet odkazů na objekt typicky
1) zvyšuje

2) snižuje
3) nemění
// 2 ok

Provedením příkazu ln -s se počet odkazů na objekt typicky
1) zvyšuje
2) snižuje
3) nemění
// 3 ok

Příznak, že objekt je symbolický odkaz, je uveden
1) v adresáři
2) v i-uzlu
3) v souboru
4) není uveden nikde
// 2 ok

Z výpisu příkazu ls -l
-rw-r--r 2 brandejs staff 0 bře 29 23:35 b
lrwxrwxrwx 1 brandejs staff 1 bře 29 23:35 c -> a
vyplývá, že
1) a je symbolický odkaz, b je tvrdý odkaz
2) c je symbolický odkaz, b je tvrdý odkaz
3) b je symbolický odkaz, a je tvrdý odkaz
4) b je symbolický odkaz, c je tvrdý odkaz
5) výpis je nesmyslný
// 2 ok

Jaká je velikost dat souboru se symbolickým odkazem ukazujícím na soubor mujprog.c
1) žádná data se k symbolickému odkazu neukládají
2) velikost dat je shodná s velikostí souboru mujprog.c
3) 0 bajtů
4) 9 bajtů
// 4 ok

Co se předá na standardní výstup po provedení posloupnosti příkazů
touch a;ln -s a b;rm a;ls
1) a
2) b
3) posloupnost příkazů je chybná, protože nelze smazat soubor, na který ukazuje symbolický odkaz
// 2 ok

Dynamickou informaci o uzamknutí souboru (výlučný přístup) obsahuje
1) i-uzel na disku
2) paměťová kopie i-uzlu
3) je uložena mimo i-uzly
// 2 ok

Informaci o přístupových právech souboru obsahuje
1) i-uzel na disku
2) výhradně paměťová kopie i-uzlu
3) je uložena mimo i-uzly
// 1 ok

Superblok je
1) oblast pro uložení dat souborů
2) informační struktura o systému souborů
3) náhradní blok za vadné sektory na disku
4) blok dat se zašifrovanou informací
// 2 ok

Speciální soubor je
1) totéž co soubor 'skrytý'
2) soubor přístupný pouze superuživateli
3) zpřístupnění v/v zařízení
4) zvláštní soubor jako adresář, symbolický odkaz apod.
5) soubor se speciálními právy
// 3 ok

Speciální soubory jsou ve výpisu příkazu ls -l identifikovány v prvním sloupci znakem nebo znaky
1) k, l
2) s
3) x
4) b, c
5) -
// 4 ok

Typickým příkladem speciálního souboru je
1) /
2) /etc/passwd
3) /var/mail/xnovak
4) /dev/null
5) /bin/ls
// 4 ok

Blokový nebo znakový typicky je
1) adresář
2) přístupový kód
3) systém souborů
4) speciální soubor
5) superblok
// 4 ok

Pojmenovaná roura
1) je způsob přenášení dat mezi dvěma počítači
2) je speciální soubor pro přenos dat mezi dvěma procesy
3) je prázdné zařízení, tzv. koš na bity
4) připojovací bod pro zpřístupnění připojeného systému souborů
// 2 ok

Která posloupnost představuje korektní vytvoření a použití pojmenované roury?
1) mkdir roura p;cat /dev/null > roura & cat roura; rm roura
2) mknod roura p;cat /etc/passwd > roura & cat roura; rm roura
3) mknod roura p;cat /etc/passwd > roura & cat roura; rmnod roura
4) mknod roura p; cat roura > /etc/passwd & cat /etc/passwd; rmnod roura
5) cat roura > /etc/passwd & mknod roura p; cat /etc/passwd; rm roura
// 2 ok

Speciální soubory pro zpřístupnění V/V zařízení se typicky ukládají do adresáře
1) /specf
2) /iofiles
3) /io
4) /dev
5) /devio

// 4 ok

Jakou bude mít velikost soubor 'x' po provedení příkazů
echo 'ahoj' > x; cp /dev/null x
1) nebude existovat
2) 0
3) 1
4) 4
5) 5
// 2 ok

Přístupová práva k objektům systému souborů se určují zvlášť pro
1) vlastníka, členy skupiny, ostatní přihlášené uživatele
2) uživatele, členy skupiny, ostatní nepřihlášené uživatele
3) uživatele, členy skupiny, ostatní přihlášené uživatele
4) vlastníka, členy skupiny, ostatní nepřihlášené uživatele
// 1 ok

Přístupová práva se nastavují v pořadí a s počátečními písmeny zkratky
1) owner, group, world
2) root, world, execute
3) user, root, others
4) user, group, others
// 4 ok

Přístupová práva k souboru se určují zvlášť pro trojici operací
1) čtení, zrušení, spuštění
2) provedení, zrušení, kopírování
3) zrušení, spuštění, editace
4) čtení, zápis, provedení
5) modifikace, zkrácení, spuštění
// 4 ok

Přístupová práva k adresáři se definují zvlášť pro trojici operací
1) vytváření, rušení, spouštění
2) vstup do adresáře, zrušení, vytvoření
3) čtení, zapisování, vstup do adresáře
4) čtení, vytváření, rušení
5) čtení, spouštění, vstup do adresáře
// 3 ok

Co znamená přístupové právo "x" pro soubor?
1) ze souboru lze číst
2) do souboru lze zapisovat
3) soubor lze spustit
4) do souboru lze vstoupit
// 3 ok

Co znamená přístupové právo "x" pro adresář?
1) adresář můžeme vypsat
2) do adresáře můžeme zapisovat
3) do adresáře je povoleno vstoupit
4) adresář je povoleno spustit
// 3 ok

Přístupová práva k souboru či adresáři jsou uložena
1) v jednom bajtu
2) ve dvou bajtech
3) na devíti bitech
4) ve dvanácti bitech
5) ve 128 bajtech
// 4 ok

Vytvořit nový soubor smím v adresáři, na který mám právo alespoň
1) čtení a zápisu
2) vytváření a vstupu
3) zápisu a vstupu
4) čtení a vytváření
5) čtení, zápisu a vstupu
// 3 ok

Zrušit soubor v adresáři smím, pokud mám alespoň
1) právo zápisu do souboru
2) právo zápisu do adresáře a zápisu do souboru
3) právo zápisu a vstupu do adresáře
4) právo zápisu a vstupu do adresáře, a právo zápisu do souboru
5) právo zápisu do souboru a jsem vlastníkem souboru
// 3 ok

Jaké právo k adresáři musím alespoň mít, pokud chci vytvořit soubor v tomto adresáři?
1) -wx
2) rw-
3) r-x
4) -w-
5) rwx
// 1 ok

V adresáři, který je na výpise příkazu ls -l označen drwxrwxrwx, smím
1) rušit libovolné soubory
2) rušit soubory, které vlastním
3) rušit soubory, ke kterým mám právo zápisu
4) nesmím rušit soubory
5) rušit soubory, které vlastním a ke kterým mám právo zápisu
// 1 ok

V adresáři, který je na výpise příkazu ls -l označen drwxrwxrwt, smím
1) rušit libovolné soubory
2) rušit soubory, které vlastním
3) rušit soubory, ke kterým mám právo zápisu
4) nesmím rušit soubory
5) rušit soubory, které vlastním a ke kterým mám právo zápisu
// 2 ok

V adresáři, který je na výpise příkazu ls -l označen d x x x, smím provést operaci:
1) rušení souboru
2) vytvoření souboru
3) nastavení adresáře jako běžného
4) výpis jeho obsahu příkazem ls
5) žádná jiná odpověď není správná
// 3 ok

Soubor mohu číst, pokud mám právo alespoň
1) čtení souboru
2) vstupu do adresáře a čtení souboru
3) čtení adresáře, vstupu do adresáře a čtení souboru
4) čtení adresáře, vstupu do adresáře a čtení souboru a vstupu do souboru
// 2 ok

Do souboru mohu zapisovat, pokud mám právo alespoň
1) zapisovat do adresáře
2) zapisovat do souboru
3) vstupovat do adresáře a zapisovat do souboru
4) zapisovat a vstupovat do adresáře, zapisovat a číst soubor
// 3 ok

Soubor (s proveditelným binárním kódem) smím spustit, pokud mám právo alespoň
1) na čtení souboru
2) čtení adresáře a spuštění souboru
3) vstup do adresáře a čtení souboru
4) vstup do adresáře a spuštění souboru
5) vstup do adresáře, spuštění souboru a čtení souboru
// 4 ok

Proces, který spustím ze spustitelného souboru obsahujícího proveditelný binární kód a majícího nastavený SUID bit, poběží vždy pod identifikací (uživatelem)
1) uživatele, který vlastní soubor
2) uživatele, který proces spustil
3) superuživatele
// 1 ok

Proces, který spustím ze spustitelného souboru obsahujícího proveditelný binární kód a nemajícího nastavený SUID bit, poběží vždy pod identifikací (uživatelem)
1) uživatele, který vlastní soubor
2) uživatele, který proces spustil
3) superuživatele
// 2 ok

Který soubor lze spustit tak, že poběží pod identifikací vlastníka souboru (vypsána jsou přístupová práva z příkazu ls -l)
1) rwxr-Sr-x
2) rws--x--x
3) rwxr-sr-x
// 2 ok

Který soubor lze spustit tak, že poběží pod identifikací uživatele, který jej spustil (vypsána jsou přístupová práva z příkazu ls -l)
1) -wS--S---
2) rws--x--x
3) rwxr-xr-x
// 3 ok

Co znamená SUID bit?
1) příznak superuživatele v souboru /etc/passwd
2) příznak vlastníka adresáře systému souborů
3) příznak měnící identifikaci majitele procesu
// 3 ok

Jsem-li vlastníkem souboru - s jakými přístupovými právy smím soubor spustit (jedná se o spustitelnou binárku; přístupová práva jsou zapsána osmičkově)?
1) 0644
2) 0766
3) 2264
4) 7666
// 2 ok

Jsem-li členem skupiny souboru, ne jeho vlastníkem - s jakými přístupovými právy smím soubor spustit (jedná se o spustitelnou binárku; přístupová práva jsou zapsána osmičkově)?
1) 0644
2) 0766
3) 2264
4) 3656
// 4 ok

Přístupová práva zapsaná osmičkově 4522 znamenají
1) r-s-w w-
2) -wS--x--x
3) rw-r--r-t
4) rwx-ws-w-
5) r--r-x-w-
// 1 ok

Přístupová práva zapsaná osmičkově 0731 znamenají
1) ---rwx-wx
2) rwxrw-r
3) rwx-wx--x
4) r-xrwx--r
5) --r-wxrwx
// 3 ok

Symbol 't' v přístupových právech znamená
1) objekty v adresáři smí rušit jen vlastník objektu nebo vlastník adresáře
2) objekty v adresáři smí rušit jen vlastník objektu
3) objekty v adresáři vytvářet jen vlastník adresáře
4) vlastníkem nově vytvořeného objektu bude vlastník adresáře
// 1 ok

Tzv. sticky bitem zapínáme následující chování
1) soubory v adresáři smí rušit pouze jejich vlastník
2) soubory v adresáři smí rušit pouze superuživatel
3) soubory v adresáři smí rušit pouze jejich vlastník a vlastník adresáře
4) soubory v adresáři smí vytvářet pouze vlastník adresáře
5) soubory v adresáři smí vytvářet pouze vlastník adresáře nebo superuživatel
// 3 ok

Jaká podmínka je postačující k tomu, abych mohl měnit přístupová práva souboru?
1) jsem vlastníkem souboru a mám právo zápisu do souboru
2) mám právo zápisu do souboru
3) mám právo zápisu do souboru a zápisu do adresáře
4) jsem vlastníkem souboru
5) mám právo zápisu do adresáře
// 4 ok

Jaká syntaxe příkazu chmod je chybná?
1) chmod go+rx
2) chmod go=o
3) chmod go+X
4) chmod u-s
5) chmod x+a
// 5 ok

Kterým příkazem přidáme ke všem položkám běžného adresáře právo zápisu pro skupinu uživatelů? (Ostatní práva neměňte.)
1) chmod g+w *
2) chmod w+g *
3) chmod w=g *
4) chmod g=w *
5) chmod w+gx *
// 1 ok

Kterým příkazem přidáme ke všem položkám běžného adresáře právo čtení pro (úplně) všechny uživatele? (Ostatní práva neměňte.)
1) chmod r=a *
2) chmod a+r *
3) chmod go+w *
4) chmod r+a *
5) chmod ugo+x *
// 2 ok

Kterým příkazem přidáte všem spustitelným souborům a všem adresářům v běžném adresáři právo spuštění/vstupu pro všechny ostatní? (Práva jiných souborů a jiná práva neměňte.)
1) chmod o=x *
2) chmod o+x *
3) chmod o+X *
4) chmod g+x *
5) nelze udělat
// 3 ok

Jakým příkazem nastavíme SUID bit?
1) chmod u+s
2) chmod g+s
3) chmod o+s
4) chmod x+s
5) chmod s+g
// 1 ok

Jakým příkazem nastavíme přístupová práva ve tvaru rw-r r
1) chmod 640
2) chmod ugo=640
3) chmod a=rw,go-w
// 3 ok

Jaká práva bude mít soubor, pokud provedu příkaz chmod 6755 soubor
1) rwsr-sr-x
2) rwsrwsr-x
3) rwsrw-rw-
4) rwxrwsr-x
// 1 ok

K čemu slouží příkaz chmod g=u
1) nastaví stejné práva pro skupinu jako má vlastník
2) nastaví stejné práva pro vlastníka jako má skupina
3) zruší všechny práva u vlastníka a skupiny
4) žádná z uvedených odpovědí
// 1 ok

Jsem členem více než jedné skupiny. Kdy má smysl, abych použil příkaz newgrp
1) chci-li vytvořit novou skupinu v systému
2) chci-li se dostat k souborům patřícím do jiné skupiny, než do které se chci přihlásit
3) chci-li při vytvoření nového souboru použít jinou skupinu, než do které jsem přihlášen
4) chci-li požádat administrátora o mé přihlášení do nové skupiny
// 3 ok

SHELL

Řídícím operátorem není
1) //
2) ||
3) &&
4) ;;
// 1 ok

Znak, který následuje za \ se
1) nepovažuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
2) nepovažuje za řídící. Je-li tím znakem "nový řádek", nepokračuje se na dalším řádku.
3) považuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
4) považuje za řídící. Je-li tím znakem "nový řádek", nepokračuje se na dalším řádku.
// 1 ok

Znaky uzavřené do dvojice apostrofů ('...') ztrácejí svůj řídící význam s výjimkou
1) ' (apostrof)
2) $, ` (obrácený apostrof), \ (následuje-li $, `, ", \, nový řádek)
3) ' (apostrof), \
4) ` (obrácený apostrof)
// 1 ok

Znaky uzavřené do dvojice uvozovek ("...") ztrácejí svůj řídící význam s výjimkou
1) ' (apostrof)
2) $, ` (obrácený apostrof), \ (následuje-li $, `, ", \, nový řádek)
3) ' (apostrof), \
4) ` (obrácený apostrof)
// 2 ok

Adresář se jménem "Ferda mravenec" (má ve jméně mezeru) nevytvořím příkazem
1) mkdir "Ferda mravenec"
2) mkdir 'Ferda mravenec'
3) mkdir `Ferda mravenec`
4) mkdir Ferda\ mravenec
5) mkdir Ferda" "mravenec
// 3 ok

Co se předá na standardní výstup po spuštění příkazu echo \'
1) Bad filename
2) echo \'
3) \'
4) ''
5) '

// 5 ok

Co se předá na standardní výstup po spustění příkazu echo \'\\\"\'
1) '\"'
2) '\\\"'
3) \\\"\'
4) \\\"
// 1 ok

Příkaz při ukončení vrací ukončovací kód. Jaký ukončovací kód je?
1) číselný; 0 znamená úspěšné ukončení, nenulové znamená ukončení s chybou
2) číselný; 1 znamená úspěšné ukončení, 0 znamená ukončení s chybou
3) číselný; nenulový znamená úspěšné ukončení, 0 znamená ukončení s chybou
4) textový; 'ok' znamená úspěšné ukončení, prázdný řetězec znamená ukončení s chybou
5) textový; prázdný řetězec znamená úspěšné ukončení, neprázdný řetězec znamená ukončení s chybou
// 1 ok

Znak \ má následující význam:
1) oddělovač adresářů a jména souboru od adresáře v cestě
2) zapnutí pokračovacího řádku
3) označení control znaku
4) zrušení řídicího charakteru následkujícího znaku
2) pokračovací řádek zapíná kombinace \ a nový řádek
// 4 ok

Znak ' (apostrof) má následující význam:
1) všechny znaky v řetězci uzavřeném do dvojice '...' ztrácí řídicí význam
2) příkaz uzavřený do dvojice '...' se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
3) znaky v řetězci uzavřeném do dvojice '...' ztrácí řídicí význam vyjma řídicích znaků $`\
4) označuje control znak
// 1 ok

Znak " má následující význam:
1) všechny znaky v řetězci uzavřeném do dvojice "..." ztrácí řídicí význam
2) příkaz uzavřený do dvojice "..." se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
3) znaky v řetězci uzavřeném do dvojice "..." ztrácí řídicí význam vyjma řídicích znaků $`\
4) označuje control znak
// 3 ok

Kolik obrácených lomítek a kolik apostrofů předá na standardní výstup příkaz echo '\'"'"\\\'
1) 2 a 1
2) 2 a 2
3) 3 a 2
4) 3 a 3
5) 4 a 4
// 2 ok

Který z příkazů nesmaže soubor x?
1) rm x
2) rm 'x'
3) rm "x"
4) rm `x`
5) rm \x
// 4 ok

Příkazsort > soubor1 < soubor2
1) čte standardní vstup ze souboru soubor2 a standardní výstup zapisuje do souboru soubor1
2) čte standardní vstup ze souboru soubor1 a standardní výstup zapisuje do souboru soubor2
3) čte standardní vstup ze souboru soubor2 a standardní chybový výstup zapisuje do souboru soubor1
4) čte standardní vstup ze souboru soubor1 a standardní chybový výstup zapisuje do souboru soubor2
// 1 ok

Operátory přesměrování jsou
1) >&, >, >>, <<-, <>
2) <<, >&, <->, <&
3) <, >, >|, <=, >>>
4) >>, <<, <$, <, >, $>
// 1 ok

Který z příkazů nezkopíruje soubor 'a' do souboru 'b'?
1) cp a b
2) cat a > b
3) cat < a > b
4) cat a|cat > b
5) cp < a > b
// 5 ok

Jaké je korektní a smysluplné přesměrování vstupu?
1) ls < adresar
2) cd < adresar
3) grep Brno < mesta
4) ls > adresar
// 3 ok

Jaké je korektní a smysluplné přesměrování výstupu?
1) echo toto se mi libi > soubor
2) cd adresar > soubor
3) mv a > b
4) grep Brno < mesta
// 1 ok

Soubor 'a' před provedením příkazu ls > a
1) musí existovat
2) nesmí existovat
3) může existovat
// 3 ok

Soubor 'a' před provedením příkazu sort < a
1) musí existovat
2) nesmí existovat
3) může existovat
// 1 ok

Po provedení příkazu
echo je > je; rm -rf neni; ls je neni > a 2> b
1) bude soubor 'a' větší než soubor 'b'

2) bude soubor 'b' větší než soubor 'a'
3) budou soubory 'a' a 'b' stejně velké
4) soubor 'b' nebude existovat nebo bude prázdný
5) soubor 'a' nebude existovat nebo bude prázdný
// 2 ok

Po provedení příkazu
echo a > b > c
1) bude soubor 'b' prázdný a soubor 'c' neprázdný
2) budou soubory 'b' a 'c' stejně velké
3) bude soubor 'b' neprázdný a soubor 'c' prázdný
4) soubor 'b' nebude existovat
5) soubor 'a' se vyprázdní
// 1 ok

Po provedení příkazu
echo b > s; echo a >> s;sort < s > s
1) bude soubor 's' prázdný
2) bude soubor 's' obsahovat řádky v pořadí 'a' a 'b'
3) bude soubor 's' obsahovat řádky v pořadí 'b' a 'a'
4) nebude soubor 's' existovat
// 1 ok protože při přesměrování výstupu se soubor vyprázdní dříve, než se provede příkaz

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
1) 5
2) 6
3) 13
4) 14
5) 15
// 5 ok

Posloupnost příkazů
cat <<ukazka
ls -l ukazka
ukazka
předá na standardní výstup
1) obsah souboru 'ukazka', výstup příkazu 'ls -l ukazka', spustí se program 'ukazka' z běžného adresáře
2) text 'ls -l ukazka'
3) text 'ukazka'
4) nepředá se nic
5) předá se výstup programu 'ukazka'
// 2 ok

Spojení příkazů do kolony je
1) ls|grep txt|sort
2) ls;grep txt;sort
3) ls&grep txt&sort
4) ls&&grep txt&&sort
// 1 ok

Návratový kód seznamurm -rf adr;mkdir adr;touch adr/so;ls adr/so|sort|grep souborbude
1) 0
2) nenulové kladné číslo
3) nenulové záporné číslo
4) prázdný řetězec
5) text s chybovým hlášením
// 2 ok

Návratový kód seznamurm -rf adr;touch adr/soubor;echo adr/soubor|sort|grep souborbude
1) 0
2) nenulové kladné číslo
3) nenulové záporné číslo
4) prázdný řetězec
5) text s chybovým hlášením
// 1 ok

Příkaz spustíme na pozadí, když
1) za příkaz napíšeme &
2) před příkaz napíšeme &
3) za příkaz napíšeme $
4) před příkaz napíšeme $
// 1 ok

Pro příkaz spuštený na pozadí neplatí
1) lze ho ukončit speciálním znakem INTR
2) lze ho ukončit příkazem kill
3) standardní vstup se napojí na /dev/null
4) byl spuštěn pomocí oddělovače &
// 1 ok

Kam je napojen standardní vstup procesu spuštěného na pozadí?
1) /dev/null
2) /dev/tty
3) /dev/zero
4) /dev/console
// 1 ok

Který z příkazů předá na standardní výstup nejprve řádek obsahující 'b' a potom řádek obsahující 'a'?
1) sleep 2;echo a&sleep 1;echo b
2) sleep 1;echo a&sleep 2;echo b
3) (sleep 1;echo a)&sleep 2;echo b
4) (sleep 2;echo a)&sleep 1;echo b
// 4 ok

Jakým způsobem nelze úlohu dostat pod správu Job Controllu?
1) příkazem fg
2) příkazem bg
3) spuštěním s &
4) speciálním znakem SUSP (^Z)
// 1 ok

Úloha běžící na pozadí se přesune na popředí příkazem
1) lg
2) kill
3) nohup
4) fg
5) žádná jiná odpověď není správná
// 4 ok

Pro seznam neplatí
1) je to posloupnost žádneho nebo více příkazu
2) příkazy jsou odděleny novým řádkem nebo středníkem (;) nebo ampersandem (&)
3) shell provádí příkazy v pořadí, v jakém jsou zapsány
4) návratový kód seznamu je návratový kód prvního provádeného procesu

5) pokud příkaz končí ampersandem (&), ihned se zahájí provádení následujiciho příkazu
// 4 ok

Oddělením dvou příkazů oddělovačem &&
1) spustíme první příkaz na popředí a druhý příkaz na pozadí
2) spustíme první příkaz na pozadí a druhý příkaz na popředí
3) provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
4) provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód
// 3 ok

K čemu slouží oddělovač && mezi dvěma příkazy?
1) druhý příkaz se provede, pokud je návratový kód prvního nenulový
2) druhý příkaz se provede, pokud je návratový kód prvního nula
3) druhý příkaz se spustí hned po spuštění prvního
4) druhý příkaz se spustí na pozadí hned po spuštění prvního
// 2 ok

Oddělením dvou příkazů oddělovačem ||
1) spustíme první příkaz na popředí a druhý příkaz na pozadí
2) spustíme první příkaz na pozadí a druhý příkaz na popředí
3) provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
4) provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód
// 4 ok

K čemu slouží oddělovač || mezi příkazy?
1) druhý příkaz se provede, pokud je návratový kód prvního nenulový
2) druhý příkaz se provede, pokud je návratový kód prvního nula
3) druhý příkaz se spustí hned po spuštění prvního
4) druhý příkaz se spustí na pozadí hned po spuštění prvního
// 1 ok

Příkazemls adresar 2>/dev/null || mkdir adresar
1) vytvoříme adresář, pokud neexistuje
2) vytvoříme adresář, pokud příkaz ls vrátil nulový návratový kód
3) vypíšeme vždy obsah adresáře a vytvoříme nový adresář
4) vypíšeme obsah adresáře
// 1 ok

Vyberte nesprávné tvrzení o seznamucd zkusto&&rm *
1) pokud adresář zkusto nebude existovat, zruší se všechny soubory běžného adresáře
2) první příkaz bude mít návratovou hodnotu 0, pokud existuje přístupný adresář zkusto
3) pokud adresář zkusto bude existovat, zruší se v něm všechny soubory
4) pokud první příkaz bude mít návratovou hodnotu 1, tak se druhý neprovede
// 1 ok

Příkazrm `cat files.txt`
1) vymaže všechny soubory, které jsou uvedeny v souboru files.txt
2) vymaže soubor files.txt
3) vymaže soubor cat i soubor files.txt
4) vymaže soubor files.txt a také soubory, které jsou v něm uvedeny
// 1 ok

Příkazrm cat files.txt
1) vymaže všechny soubory, které jsou uvedeny v souboru files.txt
2) vymaže soubor files.txt
3) vymaže soubor cat i soubor files.txt
4) vymaže soubor files.txt a také soubory, které jsou v něm uvedeny
// 3 ok

Příkazrm -rf *;touch a;echo 'ls' `ls`předá na standardní výstup
1) ls a
2) a ls
3) obsah souboru 'a'
4) 'ls' ls
5) ls `ls`
// 1 ok

Příkazecho a > seznam;rm 'seznam'smaže soubor
1) a
2) seznam
3) příkaz je chybný
4) 'a' i 'seznam'
// 2 ok

Příkazecho a > 'seznam';rm `cat seznam`smaže soubor
1) a
2) seznam
3) příkaz je chybný
4) 'a' i 'seznam'
// 1 ok

Příkazecho a > 'seznam';rm `echo seznam`smaže soubor
1) a
2) seznam
3) příkaz je chybný
4) 'a' i 'seznam'
// 2 ok

Příkazecho a > `seznam`;rm 'cat seznam'smaže soubor
1) a
2) seznam
3) příkaz je chybný
4) 'a' i 'seznam'
// 3 ok

Proměnná se v shellu "deklaruje" příkazem
1) var ADRESAR=/tmp
2) ADRESAR=/tmp
3) $ADRESAR=/tmp
4) strings $ADRESAR /tmp
// 2 ok

Jak vytvořím proměnou ADRESAR obsahující slovo EMPTY
1) ADRESAR=EMPTY
2) ADRESAR:=EMPTY
3) $ADRESAR=EMPTY
4) touch ADRESAR < EMPTY
// 1 ok

Zadali jsmevelikost=maxiJak vypíšeme řetězec "maxipes"?
1) echo $velikost."pes"
2) echo $[velikost]pes
3) echo $velikostpes
4) echo ${velikost}pes
// 4 ok

Deklarace proměnné spolu s jejím naplněním se provádí
1) jmeno_promenne=[hodnota]
2) jmeno_promenne:=[hodnota]
3) jmeno_promenne<[hodnota]
4) $jmeno_promenne:=[hodnota]
// 1 ok

Uživatelem zavedené proměnné se v shellu dědí do potomků
1) vždy
2) nikdy
3) jen proměnné označené příkazem export
4) jen proměnné označené příkazem import
5) jen proměnné označené příkazem child
// 3 ok

Jaké je nesprávné použití obsahu proměnné ADRESAR?
1) echo $ADRESAR/NEW
2) echo ${ADRESAR}NEW
3) echo $ADRESAR NEW
4) echo $ADRESARNEW
// 4 ok

Co provádí příkaz set spuštěný bez voleb a argumentů?
1) vyprázdní všechny proměnné
2) smaže všechny proměnné
3) vypíše všechny proměnné
4) exportuje všechny proměnné
// 3 ok

Ve kterém způsobu závorkování se provede příkaz ve vnořeném shellu?
1) (prikaz)
2) { prikaz;}
3) <prikaz;>
// 1 ok

Příkazcd `echo $OLDPWD`se chová stejně jako:
1) pwd
2) cd ~
3) cd -
4) cd ..
// 3 ok

Který z příkazů má správnou syntaxi?
1) { echo ahoj;}
2) { echo ahoj }
3) {echo ahoj};
4) {echo ahoj}
5) {echo ahoj }
// 1 ok

Máme nastavený pracovní adresář /tmp/data. Které z použití příkazu pwd vypíše jiný pracovní adresář než /tmp/data?
1) cd $PWD;pwd
2) { cd $PWD;};pwd
3) { cd $HOME;};cd -;pwd
4) { cd $HOME;};pwd
5) (cd $HOME);pwd
// 4 ok

Co je uloženo v proměnné PWD
1) pracovní adresář
2) heslo, zašifrované jednosměrnou šifrou
3) domovský adresář
4) hesla všech uživatelů (zašifrovaná jednosměrnou šifrou)
// 1 ok

Mezi složené příkazy nepatří
1) for
2) case
3) while
4) until
5) find
// 5 ok

Co vykonává příkaz true?
1) vrací hodnotu true
2) vypisuje řetězec true dokud není ukončen
3) vrací návratový kód 0
4) vrací návratový kód 1
5) není to příkaz, je to logická hodnota
// 3 ok

[je
1) příkaz
2) otevírací závorka pro zápis podmínky
3) otevírací závorka pro zápis osmičkového čísla
4) symbol přesměrování
5) speciální znak
// 1 ok

Který z příkazů nesmaže všechny soubory z běžného adresáře?
1) for S in *; do rm $S; done
2) rm `ls`
3) rm `echo *`
4) for S in `ls`; do rm $S; done
5) všechny ostatní příkazy soubory smažou
// 5 ok

Kterým příkazem pošleme správně celý text e-mailem na zadanou adresu?
1) echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík>mail adresa@nekde
2) (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)>mail adresa@nekde
3) (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)|mail adresa@nekde
4) echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík|mail adresa@nekde
// 3 ok

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?
1) for JM in $PWD; do echo $JM ${JM}.o; done
2) for $JM in *; do echo $JM $JM.o; done
3) for JM in *; do echo $JM $JM.o; done

4) for JM in *; do echo JM JM.o; done
// 3 ok

Jaká je korektní syntaxe zápisu vyhodnocení podmínky?
1) if `ls soubor`; then rm soubor; fi
2) if 'ls soubor'; then rm soubor; fi
3) if ls soubor; then rm soubor; fi
4) if [ls soubor]; then rm soubor; fi
// 3 ok

Kterým příkazem nezjistím, zda položka 'adresar' je adresář?
1) if test -d adresar; then echo ok; fi
2) if [-d adresar]; then echo ok; fi
3) if ls -ld adresar|grep -q '^d'; then echo ok; fi
4) if pwd adresar; then echo ok; fi
// 4 ok

Který příkaz je syntakticky špatně?
1) if [$# -ge 3] && [$# -lt 6]; then echo ok; fi
2) if [$# -ge 3 && $# -lt 6]; then echo ok; fi
3) if [$# -ge 3 -a $# -lt 6]; then echo ok; fi
// 2 ok

Při kterém volání skriptu
#!/bin/bash
if [$# = 4 -a $1 != $2]; then echo ok; fi
se předá na standardní výstup 'ok'?
1) ./skript 'a' `echo a` a A
2) ./skript 1 2 '' 3 4
3) ./skript 1 2 '' 4
4) ./skript a \a b \b
// 3 ok

Který soubor nevypíše příkazls x?[a-c]*
1) žádný soubor, který by měl jméno delší než 4 znaky
2) soubor x1.c i když se v adresáři nachází
3) soubor xabc i když se v adresáři nachází
4) soubor x?abc* i když se v adresáři nachází
// 2 ok

Jaká je správná syntaxe dotazu na existenci souboru?
1) if [-f soubor] then echo ano fi
2) if [-f soubor] then; echo ano fi;
3) if [-f soubor]; then echo ano; fi
4) if [-f soubor;]; then echo; ano; fi
5) if [-f soubor;]; then echo ano; fi;
// 3 ok

Který příkaz je syntakticky správně a současně obsahuje co nejméně mezer?
1) [-d adresar]&&echo ano
2) [-d adresar]&&echo ano
3) [-d adresar]&&echo ano
4) [-d adresar] &&echo ano
5) [-d adresar] && echo ano
// 3 ok

Pokud proměnná A obsahuje číslo větší než 5, který příkaz podmínku A > 5 korektně vyhodnotí a předá na standardní výstup 'ano'?
1) if [$A > 5] then echo ano fi;
2) if [$A > 5]; then echo ano; fi
3) if [$A -gt 5] then echo ano; fi;
4) if [$A -gt 5]; then echo ano; fi
// 4 ok

Kterým/i znakem/znaky začínají v shellu komentáře?
1) #
2) //
3) !
4) %
5) /*
// 1 ok

Kterým příkazem spustím skript v běžném shellu?
1) /skript
2) .skript
3) . skript
4) ! skript
5) žádná jiná odpověď není správná
// 3 ok

Chci-li spustit skript příkazem ./skript, musím mít na soubor se skriptem právo minimálně
1) spuštění
2) čtení a spuštění
3) čtení, zápis a spuštění
4) čtení
5) zápis a spuštění
// 2 ok

Mějme skript
#!/bin/sh
echo $4$1$3$2
Co předá na standardní výstup příkaz ./skript a "" b f
1) f a b
2) fab
3) af b
4) afb
5) b a f
// 2 ok

Co předá na standardní výstup příkaz "echo $2", jestliže byl předtím proveden příkaz "set a b c"?
1)
2) a
3) b
4) c
// 3 ok

Co předá na standardní výstup příkaz(exit 1; exit 0); echo $?
1) nic, shell se ukončí
2) 0
3) 1
4) ?
// 3 ok

Pracovní soubor v adresáři /tmp s unikátním jménem (žádný jiný aktivní proces spuštěný z téhož skriptu nepoužije stejné jméno) vytvořím příkazem
1) touch /tmp/pomocny
2) touch /tmp/pomocny.$?
3) touch /tmp/pomocny.$$
4) touch /tmp/pomocny.$%
5) touch /tmp/pomocny.$!
// 3 ok

Příkazls ~předá na standardní výstup položky z
1) běžného adresář
2) domovského adresáře
3) kořenového adresáře
4) adresáře /tmp
5) adresáře /bin
// 2 ok

V běžném adresáři máte
položky
a
aa
amanda
am da (mezera ve jméně)
AMANDA
Která jména předá na standardní výstup příkaz ls a*a
1) aa amanda am da
2) aa amanda
3) amanda
4) aa amanda am da AMANDA
5) a aa amanda am da AMANDA
// 1 ok

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]
1) a
2) aa
3) amanda am da
4) amanda am da AMANDA
5) [a-m][!0-9]
// 2 ok

Skript, který se spustí poté, co se přihlásím do systému, je uložen v souboru
1) /home/.profile
2) /home/profile
3) ~/.profile
4) ~/profile
5) /etc/login
// 3 ok

Mám v běžném adresáři soubory .kshrc, dopis.txt, a.out, prog.c. Co předá na standardní výstup příkaz 'echo *' ?
1) *
2) .kshrc dopis.txt a.out prog.c
3) dopis.txt a.out prog.c .kshrc
4) .kshrc
5) a.out dopis.txt prog.c
// 5 ok

Mám v běžném adresáři jména souborů a, a1, aa1, 1a
Která jména souborů předá na standardní výstup příkaz 'ls a[a1]*'?
1) a a1 aa1
2) aa1 a1
3) 1a a a1 aa1
4) aa1 a1 1a
// 2 ok

Který příkaz předá na standardní výstup číslo 4?
1) echo 2+2
2) echo A:=2+2;
3) echo `(2+2)`
4) echo $[2+2]
5) echo $2+$2
// 4 ok

Příkaz A=5;B=2;echo C=$A+$B předá na standardní výstup
1) 7
2) C=7
3) C=5+2
4) C=A+B
5) nic
// 3 ok

Který skript se spouští vždy při spuštění nového shellu bash?
1) ~/.profile
2) ~/.bashrc
3) /bash/bashrc
4) /bin/bash/profile
// 2 ok

Jakým příkazem předám na standardní výstup hodnotu prvního pozičního parametru?
1) echo $1
2) cat $1
3) $1
4) set $1
5) cat '$1'
// 1 ok

Vytvořte adresář zadaný prvním pozičním parametrem. Který příkaz jej nevytvoří?
1) if [! -d $1]; then mkdir $1; fi
2) test -d $1 || mkdir $1
3) [! -d $1] && mkdir $1
4) if test ! -d $1; then mkdir $1; fi
5) [-d $1] && mkdir $1
// 5 ok

Jakým příkazem spustíte skript v běžném adresáři a jako první pozční parametr mu předáte řetězec ahoj?
1) ./skript 1=ahoj
2) ./skript $1=ahoj
3) $1=ahoj; ./skript
4) ./skript;set ahoj
5) ./skript ahoj
// 5 ok

Kterým příkazem smažu všechny soubory zadané pozičními parametry (pozičních parametrů může být libovolné množství)?
1) rm $*
2) rm `$*`
3) rm *
4) rm $1 $2 $3 $4 ...
5) rm `*`
// 1 ok

Jak nespustíme skript, abychom mu jako poziční parametry předali jména položek běžného adresáře začínající písmenem A?
1) ./skript `echo A*`
2) ./skript 'ls A*'
3) ./skript A*
// 2 ok

Příkaz echo $$ předá na standardní výstup
1) jméno spuštěného skriptu
2) číslo procesu shellu
3) počet pozičních parametrů předaných skriptu
4) všechny poziční parametry předané skriptu
// 2 ok

Jaká minimální oprávnění jsou nutná pro spuštění shellového skriptu příkazem bash skript
1) právo spuštění/provedení
2) právo čtení
3) právo spuštění/provedení a právo čtení
4) právo spuštění/provedení a právo zápisu
5) právo spuštění/provedení a právo čtení a právo zápisu
// 2 ok

Jaká je správná syntaxe příkazu pro porovnání numerického obsahu dvou proměnných?
1) [$A > $B]
2) [$A -gt $B]
3) [$A>$B]
4) [$A-gt$B]
// 2 ok

Příkaz touch knihovna; test -d knihovna na jinak prázdném adresáři vrátí návratový kód
1) prázdný
2) 0
3) 1
// 3 ok

Příkazem PS1=ferda změníme
1) implicitní podpis v e-mailu
2) nastavení běžného adresáře
3) výzvu na terminálu uživatele
4) jméno mailboxu
5) primární aktivní proces
// 3 ok

Příkazovou substituci uzavíráme do
1) '...'
2) "..."
3) `...`
4) <...>
// 3 ok

Operátorem 2>
1) přesměrujeme standardní výstup
2) přesměrujeme standardní chybový výstup
3) přesměrujeme standardní vstup
4) zavřeme standardní výstup
5) zrušíme chybový výstup
// 2 ok

Kterým příkazem vypíšeme na standardní výstup jména všech adresářových položek, které kdekoli ve svém jméně mají písmeno A?
1) echo .*A.*
2) ls '.*A*'
3) echo *A*
4) ls ".A.*"
5) ls *A.*
// 3 ok

Jakým příkazem spojíme za sebe obsah souborů a a b a výsledek uložíme do souboru c?
1) cat < a < b > c
2) cat a b | c
3) cat a b > c
4) cat < a < b | c
5) a b | cat > c
// 3 ok

Vytvořte adresář Mail v domovském adresáři uživatele, pokud tam již není. Jaký příkaz toto neprovede?
1) [-d ~/Mail] || mkdir ~/Mail
2) (cd; test -d Mail || mkdir Mail)
3) if [! -d $HOME/Mail]; then mkdir $HOME/Mail; fi
4) (cd ~; mkdir Mail 2>/dev/null)
5) test -d ~/Mail && mkdir Mail 2>/dev/null
// 5 ok

Jakým příkazem zkopírujete obsah souboru a do souborů b, c, d současně?
1) cat a > b > c > d
2) cp a b c d
3) tee < a > b > c > d
4) cat a | tee b c > d
5) tee a | cat b c d
// 4 ok

Jakým příkazem přidáte za konec souboru x řádek obsahující xxx-konec-xxx?
1) echo xxx-konec-xxx >> x
2) cat x < xxx-konec-xxx
3) cat x|echo xxx-konec-xxx > x
4) cat xxx-konec-xxx << x
// 1 ok

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.
1) for S in *.txt; do echo $S; done
2) cat *.txt
3) echo *.txt
4) ls -l *.txt
// 3 ok

REGEXP

V textovém souboru jsou řádky obsahující takto uspořádané informace:
Michal Brandejs <brandejs>
Jakým substitučním příkazem editoru vi je všechny hromadně převedete do tvaru:
Jmeno: Michal Prijmeni: Brandejs e-mail: brandejs
1) řešení: 1,$s/^[A-Za-z]*[ ][A-Za-z]*[<][a-z]*[>]/Jmeno: \1 Prijmeni: \3 e-mail: \5/
2) řešení: 1,$s/\(^[A-Za-z]*\)[ ]\([A-Za-z]*\)[ ]\(<[a-z]*>\)/Jmeno: \1 Prijmeni: \2 e-mail: \3/

3) řešení: 1,$s/\(.*\) \(.*\) <\(.*\)>/Jmeno: \1 Prijmeni: \2 e-mail: \3/
4) řešení: 1,$s/^[A-Za-z]*[ ][A-Za-z]*[<][a-z]*[>]/Jmeno: $1 Prijmeni: $3 e-mail: $5/
// 3 ok

V textovém souboru jsou řádky obsahující takto uspořádané informace:
brandejs:*:11000:100:Michal Brandejs:/home/brandejs:/bin/ksh
Jakým substitučním příkazem editoru vi je všechny hromadně převedete do tvaru:

Michal Brandejs <brandejs>

1) řešení: 1,$s/(.*):.*:.*:.*:(.*):.*/\2 <\1>/
2) řešení: 1,$s/\(.*\):.*:.*:.*:\(.*\):.*:.*/\2 <\1>/
3) řešení: 1,$s/\([^:]*:\).*:\([^:]:\).*/\2 <\1>/
// 2 ok

Regulární výraz [0-9][^0-9].x vyhovuje např. řetězci
1) 0ax
2) a0x
3) 00bx
4) 0abx
// 4 ok

Jakým substitučním příkazem editoru vi nahradíme řetězec \end{syntax} řetězcem \end{syntax}% (stačí provést 1x na řádku)?
1) 1,$s/\end\{syntax\}/\end\{syntax\}%/
2) 1,$s/\end{syntax}/\end{syntax}%/
3) 1,$s/\\end{syntax}/\\end{syntax}%/
4) 1,$s/\\\end{syntax}/\\\end{syntax}%/
// 3 ok

Kterým substitučním příkazem editoru vi zrušíte (vyprázdníte) na běžném řádku všechny řetězce znaků začínající rovnítkem, následované alespoň jednou mezerou, přičemž zrušit se musí všechny mezery za rovnítkem?
1) s/=[ ][ ]*//g
2) s/=[ ]*//g
3) s/=[ ][ ][ ]*//g
4) s/=*//g
5) s/=*[ ]//g
// 1 ok

Kterým substitučním příkazem editoru vi v celém souboru vyprázdníme všechny řádky, které obsahují pouze řetězec ahoj?
1) 1,$s/ahoj//g
2) 1,$s/*ahoj*//g
3) 1,$s/[^]*ahoj[^]*//
4) 1,$s/^ahoj$//
5) 1,$s/\^ahoj\$//
// 4 ok

Jakým substitučním příkazem editoru vi zrušíme nadbytečné mezery na běžném řádku (tzn. řetězce mezer delší než 1 znak nahradit jednou mezerou)?
1) s/[ ]*/ /
2) s/ [ ]*/[ ]/
3) s/[ ][ ]*/ /
4) s/[ ][ ]*/[ ]/
// 3 ok

Jakým substitučním příkazem editoru vi přidáme za poslední znak každého řádku znak středník?
1) 1,$s/.*/;/
2) 1,$s/.*$/;/
3) 1,$s/$/;/
4) 1,$s/^.*/;/
5) 1,$s/$*/;/
// 3 ok

V souboru máte vždy na jednom řádku jedno URL. Jakým příkazem editoru vi zrušíte z konců všech řádků řetězec znaků index.htm nebo index.html? Tzn. řádky obsahující např. http://www.fi.muni.cz/index.html změňte na http://www.fi.muni.cz/. Za URL bezprostředně následuje znak nového řádku.
1) 1,$s/index\.html\{0,1\}$//
2) 1,$s/[a-z]*$//
3) 1,$s/\/index.html*//
4) 1,$s/\/[a-z].[a-z]$//
// 1 ok

Jakým substitučním příkazem řádkového režimu editoru vi změníte všechny řádky souboru ze tvaru
Lampa stolní typ GY34827|B240|3487636|790
do tvaru
Lampa stolní typ GY34827|B240|DKP 3487636|790 Kč
1) 1,$s/\(.*\)|\(.*\)|\(.*\)|\(.*\)/\1DKP \2 Kč/
2) 1,$s/\(.*|.*|\)\(.*|.*\)/\1DKP \2 Kč/
3) 1,$s/\(.*\)|\(.*\)|[0-9]*|\(.*\)/\1DKP \2 Kč/
// 2 ok

TEXT

Jakým příkazem z proměnné PATH vyřadíme adresář /bin ? Pro jednoduchost předpokládejte, že není ani na začátku, ani na konci.
1) PATH=`echo $PATH|sed 's/:\/bin:/:/'`
2) PATH=echo $PATH|`sed 's/:\/bin:/:/'`
3) PATH=`sed 's/:\/bin:/:/'`
// 1 ok

Zda soubor file obsahuje textový řetězec string zjistíme příkazem
1) find string file
2) fgrep string file
3) sed string file
4) string string file
5) search string file
// 2 ok

Počet adresářových položek v běžném adresáři zjistíme příkazem
1) ls|wc -l
2) ls -l
3) count `ls`

// 1 ok

Soubory obsahující v názvu podřetězec '_delete' v běžném adresáři a všech jeho podadresářích smažeme příkazem
1) for S in *_delete*; do rm $S; done
2) find . -name '*_delete*' -exec rm {} \;
3) rm *_delete*
4) for SO in `ls *_delete*`; do rm $SO; done
5) for SO in `ls _delete`; do rm $SO; done
// 2 ok

Který příkaz ze všech souborů v běžném adresáři vyřadí řádky, které v prvním sloupci začínají znakem '#'?
1) grep -v \^\# *
2) for S in *; do grep -v \# $S > $S; done
3) for S in *; do grep -v '^#' $S > $S.s; mv -f $S.s $S; done
4) find ~ -type f -exec grep -v \^\# {} | dd of={} \;
// 3 ok

Kterým příkazem smažeme (pouze) v běžném adresáři soubory obsahující ve svém jméně podřetězec 'tempor'
1) for S in *; do grep -q tempor $S && rm $S; done
2) find . -name '*tempor*' -exec rm {} \;
3) find . -type f -exec grep -q tempor {} \; -exec rm {} \;
4) rm *tempor*
// 4 ok

Kterým příkazem předáme na standardní výstup cesty k souborům, které buď ve jméně souboru, nebo ve svém obsahu mají podřetězec 'brno'? Soubory hledejte ve svém domovském adresáři a ve všech jeho podadresářích.
1) find ~ -type f \(-name '*brno*' -or -exec grep -q brno {} \; \) -print
2) find ~ -type f -exec grep -q brno {} \; -name '*brno*' -print
3) find ~ -type f (-name brno -or -exec grep -q '*brno*' {} \;) -print
4) find ~ -type f \(-name '*brno*' -or -exec grep -q '*brno*' {} \; \) -print
// 1 ok

Kterým příkazem předáme obsah souboru so na standardní výstup?
1) echo so
2) cat so
3) grep so
4) print so
// 2 ok

Jakým příkazem předáme na standardní výstup řetězec ahoj?
1) echo ahoj
2) cat ahoj
3) grep ahoj
4) cat '*ahoj*'
5) echo [ahoj]
// 1 ok

Jakým příkazem předáme na standardní výstup počet adresářových položek běžného adresáře zvýšený o 2?
1) cat $(`ls|wc -`+2)
2) $((`ls|wc -`+2))
3) wc -l $((`ls`+2))
4) echo $[`ls|wc -l`+2]
5) for SO in *; do echo 2;done|wc -l
// 4 ok

Jakým příkazem zrušíme všechny podadresáře v běžném adresáři, které obsahují více než 10 položek?
1) for S in *; do [-d $S -a `find $S -print|wc -l` -gt 10] && rm -rf $S; done
2) find . -type d -exec [ls {}|wc -l] > 10 \; -exec rm -rf {} \;
3) find . -type d -exec for S in {}; do [ls -R|wc -l -gt 10]; done \; && rm -rf {} \;
4) for S in *; do test -d $S && test wc -l `ls -R` -gt 10 && rm -rf $S; done
// 1 ok

Jakým příkazem přejmenujeme všechny soubory v běžném adresáři mající příponu .txt na příponu .tex?
1) mv *.txt *.tex
2) for S in *.txt; do N=`echo $S|sed 's/\.txt$/.tex/'`; mv $S $N; done
3) find . -type f -name '*.txt' -exec mv {}.txt {}.tex \;
4) find . -type f -name '*.txt' -exec mv {} ${%%.txt}.tex \;
5) rm *.txt *.tex
// 2 ok

Do souboru b zkopírujte ty řádky ze souboru a, které obsahují symbol > (větší než). Jakým příkazem provedete toto zadání?
1) cat a|sed s/>// > b
2) grep > a '>' b
3) grep '>' a > b
4) sed s/>// < a > b
// 3 ok

Jakým příkazem předáte na standardní výstup jména všech souborů běžného adresáře, které ve svém těle obsahují textový řetězec Brno?
1) ls *Brno*
2) grep *Brno* *
3) grep Brno *
4) find . -type f -exec grep *Brno* {} /dev/null \;
5) echo *Brno*
// 3 ok

Jakým příkazem neporovnáme např. shodu dvou řetězců, chceme-li je zadat jako argumenty na příkazovém řádku?
1) [
2) cmp
3) test
// 2 ok