Operační systém UNIX vznikl okolo roku
• 1950
• 1960
• **1970**
• 1980
• 1990
--
První verze UNIXu byla naprogramována pro počítač
• IBM PC
• **PDP**
• IBM 370
• VAX
• Sun
--
První verze UNIXu byla vytvořena v
• **Bell Laboratories**
• University of Berkeley
• Microsoftu
• Novellu
• USL (UNIX System Laboratories)
--
Co vzniklo dříve? MS-DOS (předchůdce MS-Windows) nebo UNIX?
• MS-DOS
• **UNIX**
• Obojí je zhruba stejně staré.
--
Kdo jsou autoři prvních verzí UNIXu?
• Steeve Jobs, Tim O'Reilly, Andrew Tanenbaum
• Larry Ellison, Scott McNealy, Michael Tiemann
• **Brian Kernighan, Dennis Ritchie, Ken Thompson**
• Eric Raymond, Richard Stallman, Linus Torvalds
--
Operační systém Linux vzniká od roku
• 1951
• 1961
• 1971
• 1981
• **1991**
--
Záznamy o uživatelích, kteří mají právo přistupovat k systému, jsou uloženy v souboru:
• /etc/group
• /etc/users
• **/etc/passwd**
• /etc/shadow
• /etc/motd
--
Hesla uživatelů v otevřeném tvaru jsou uložena v uživatelům nepřístupném souboru:
• /etc/passwd
• /etc/passwords
• /etc/shadow
• **nejsou uložena**
--
Soubor /etc/passwd v aktuálních verzích systému
• obsahuje zašifrovaná hesla uživatelů
• **neobsahuje hesla uživatelů**
• nikdy, ani v minulosti neobsahoval hesla uživatelů
• obsahuje nezašifrovaná hesla uživatelů
--
Účet uživatele **ne**obsahuje
• primární skupinu
• uživatelské jméno
• UID uživatele
• **UČO uživatele**
• žádná z ostatních odpovědí není správná
--
Hesla uživatelů v zašifrovaném tvaru jsou uložena v uživatelům
nepřístupném souboru:
• /etc/passwd
• **/etc/shadow**
• /etc/passwords
• nejsou uložena
--
Uživatel zapomněl svoje heslo. Jak jej nejsnáze získá?
• uživatel použije příkaz decryptpas
• uživatel požádá superuživatele o provedení příkazu decryptpas
• uživatel požádá superuživatele o získání svého hesla ze souboru /etc/shadow
• **žádná jiná odpověď není správná**
--
Uživatelské jméno se nazývá:
• **login**
• loggin
• user name
• uid
• user

--
Hodnota UID u uživatele root je
• A
• root
• 0
• 1
• 99999
--
Uživatel je členem
• nemusí být členem žádné skupiny
• je členem alespoň jedné skupiny
• je členem alespoň dvou skupin
--
Seznam skupin uživatelů v systému je uložen v souboru
• /etc/team
• /etc/groups
• /etc/group
• /etc/teams
• /etc/cluster
--
Soubor /etc/passwd musí být
• čitelný pouze superuživateli
• nesmí být čitelný nikomu
• čitelný pro všechny
• čitelný pouze skupině root
• žádná jiná odpověď není správná
--
Soubor /etc/shadow musí být
• čitelný pouze superuživateli
• nesmí být čitelný nikomu
• čitelný pro všechny
• žádná jiná odpověď není správná
--
Každé uživatelské jméno v souboru /etc/passwd musí být definováno
• alespoň jednou
• právě jednou
• nemusí být definováno vůbec
• právě dvakrát
• alespoň dvakrát
--
Každé uživatelské jméno v souboru /etc/group musí být použito
• alespoň jednou
• právě jednou
• nemusí být uvedeno vůbec
• právě dvakrát
• alespoň dvakrát
--
Interpret příkazů a domácí adresář, který se spustí a nastaví po přihlášení uživatele, je popsán
• v souboru .profile
• v účtu uživatele
• stavem při posledním odhlášení
• v souboru /etc/userdef
• v UID záznamu
--
Čas v unixu je uložen (např. čas změny obsahu souboru)
• ve tvaru yyyymmddhhmiss
• ve tvaru ddmmyyyyhhmiss
• v počtu hodin od začátku epochy
• v počtu minut od začátku epochy
• v počtu sekund od začátku epochy
--
Epocha v unixu začala
• 1. 1. 1970 00:00
• 31. 12. 1970 24:00
• 1. 1. 1980 00:00
• 31. 12. 1980 24:00
--
Počet údajů na řádku v souboru /etc/passwd je (některý může být nevýznamný) alespoň
• 4
• 5
• 6
• 7
• 8
--
Odlišnosti v chování textových terminálů jsou
• sjednoceny instalací kompatibilního driveru
• transformovány na jednotné volání popisem v souboru /etc/termcap
• eliminovány společným voláním přes databázi v souboru /etc/termdbf
• omezeny na kompatibilní typy terminálů v compatibility seznamu
--
Démon je
• je ovladač spících zařízení s nejnižší prioritou
• je spící program aktivovaný událostí

• infiltrovaný interpret příkazů
• ladící systém jádra operačního systému
--
Každý řádek textového souboru v unixu končí
• znakem CR
• znakem LF
• znaky CR,LF
• znaky LF,CR
--
Každý řádek textového souboru v unixu končí
• znakem Carriage Return
• znakem Long Field
• znaky CR,LF
• znaky LF,CR
• žádná z ostatních odpovědí není správná
--
Každý řádek textového souboru v MS-DOSu/Windows končí
• znakem CR
• znakem LF
• znaky CR,LF
• znaky LF,CR
--
Při přenosu textového souboru z MS-DOSu/Windows do unixu
• přidáváme řídící znak CR za každý řádek
• přidáváme řídící znak LF za každý řádek
• odebíráme řídící znak CR za každým řádkem
• odebíráme řídící znak LF za každým řádkem
--
Při přenosu textového souboru z unixu do MS-DOSu/Windows
• přidáváme řídící znak CR za každý řádek
• přidáváme řídící znak LF za každý řádek
• odebíráme řídící znak CR za každým řádkem
• odebíráme řídící znak LF za každým řádkem
--
Primární prompt běžného uživatele unixu je znak
• #
• $
• &
• *
• %
--
Primární prompt superuživatele je znak
• #
• $
• &
• *
• %
--
Primární prompt si uživatel nastavuje/mění v proměnné prostředí
• PROMPT
• PRMPT
• PRM1
• PS1
• PS
--
Příkazem `stty -a`
• se uživatel odhlásí ze sezení u terminálu
• se uživateli obnoví výchozí nastavení terminálu
• se uživateli sdělí nastavení terminálu
• se uživateli zpřístupní všechny terminály
• se uživateli přepne terminál do ascii režimu
--
Konvence pro prezentaci (výpis) znaku control-c je
• CTRL-C
• CC
• ^C
• C-C
• #C
--
Výpis (prezentace) \012 znamená
• zápis hodnoty -12
• znak s ordinální hodnotou 12 dekadicky
• znak s ordinální hodnotou 12 oktalově
• znak s ordinální hodnotou 12 hexadecimálně
• textový řetězec 12
--
Speciální znak **intr** (typicky control-c)
• přeruší činnost operačního systému
• přeruší běh procesu na popředí
• přeruší výpis na terminál na popředí
• přeruší výpis na terminál na pozadí
--
Proces běžící na popředí násilně ukončím (např. pokud cyklí v nekonečné smyčce) stiskem speciálního znaku

• intr
• stop
• kill
• eof
--
Speciální znak **eof** (typicky control-d)
• ukončí stav X-off
• přeruší běh procesu
• ukončí zpracování fail stavu
• vloží příznak konce souboru
--
Speciální znaky **start** a **stop** (typicky control-q a
control-s)
• realizují protokol X-ON/X-OFF
• spustí a ukončí proces
• přihlásí a odhlásí uživatele
• spustí a ukončí komunikaci operačního systému
--
Při provádění příkazu `cp /dev/tty /dev/null` **ne**získáte prompt stisknutím speciálního znaku
• stop
• eof
• intr
• susp
--
Co provede příkaz `cp /etc/passwd /dev/tty`
• zkopíruje obsah běžného souboru /etc/passwd do běžného souboru /dev/tty
• zkopíruje obsah běžného souboru /etc/passwd na první tiskárnu systému
• zkopíruje obsah běžného souboru /etc/passwd na terminál uživatele
• čte z klávesnice terminálu a zapisuje do souboru /etc/passwd
--
Přihlašovací (login) shell **nelze** za žádných okolností ukončit
• speciálním znakem eof
• speciálním znakem stop
• speciálním znakem intr
• příkazem exit
• rozpadnutím spojení
--
Která služba používá šifrovanou komunikaci mezi klientem a serverem?
• telnet
• rsh
• ssh
• ftp
--
Jaký počet kořenových adresářů je uživateli dostupný?
• tolik, kolik je v systému připojených diskových zařízení
• tolik, kolik je v systému připojených diskových zařízení, plus 1
• právě jeden
• žádný
--
Při úspěšném přihlášení uživatele do systému se **ne**provede
• vypíší se systémové zprávy, jsou-li nějaké
• vypíše se sdělení správce systému, je-li nějaké
• spustí se uživatelem vybraný shell dle obsahu /etc/passwd
• uživatel se přihlásí do skupiny dle obsahu /etc/group
• vypíše se prompt shellu
--
Při zadávání příkazů shellu **ne**platí, že
• lze příkazy zadávat "do zásoby"
• lze u některých shellů využít paměti starých (historii) příkazů
• příkazy lze zadávat bez ohledu na velikost písmen
• po stisknutí klávesy Enter již nelze opravovat obsah odeslaného řádku
• řádek lze upravovat tak dlouho, dokud není stisknuta klávesa Enter
--
Při zadávání příkazů shellu klávesou Enter ještě **ne**odešleme příkaz shellu ke zpracování (použijeme pokračovací řádek), když
• klávesu Enter stiskneme na konci řádku
• klávesu Enter stiskneme rychle dvakrát po sobě (tzv. doubleclick)
• před stiskem klávesy Enter vložíme obrácené lomítko
• po stisku klávesy Enter vložíme znak větší-než
--
Sekundární prompt se používá
• ve vnořeném shellu
• na pokračovacím řádku
• při čtení příkazů ze skriptu
• na řádku zrušeném speciálním znakem kill
--
Sekundární prompt lze přenastavit změnou proměnné prostředí PS2
a implicitně obsahuje znak
• $
• >
• #
• &
• *

--
Historii mnou zadaných příkazů shellu bash mi vypíše příkaz
• man history
• cat history
• cat ~/.history
• cat ~/.bash_history
• žádná z ostatních odpovědí není správně
--
Historie zadaných příkazů shellu bash se ukládá do souboru s historií příkazů, když
• se uživatel přihlašuje
• uživatel zadává jakýkoli příkaz
• se uživatel odhlašuje
• uživatel zadá příkaz history
--
Operační systém Linux
• vznikl předáním ochranné známky UNIX společnosti GNU
• je předchůdcem unixu
• vznikl nezávisle na unixu
• je následovníkem unixu
--
Uživatel ke v/v zařízením v unixu přistupuje prostřednictvím
• instalovaných driverů
• ikon zařízení
• speciálních souborů
• přerušení
--
UNIX byl při svém vzniku koncipován jako systém
• multiprogramový, multiuživatelský, sí»ový
• multiprogramový, multiuživatelský, s terminálovým přístupem
• multiprogramový, sí»ový, s terminálovým přístupem
• multiuživatelský, sí»ový, s terminálovým přístupem
--
Ukončí-li se shell, který byl spuštěn po přihlášení uživatele
• je uživatel odhlášen ze systému
• automaticky se spustí další shell, není-li zadán příkaz exit
• tento shell uživatel nemůže ukončit, protože je systémový
• systém se uživatele zeptá, zda chce spustit novou kopii shellu
--
Bezprostředně po ukončení právě prováděného příkazu shellu, jsou-li zadány nějaké příkazy "do zásoby"
• se příkazy v zásobě ihned provedou bez potvrzení uživatele
• musí uživatel potvrdit provedení každého příkazu v zásobě zvláš»
• musí uživatel potvrdit celou skupinu příkazů v zásobě, jsou provedeny bud všechny nebo ani jeden
• nelze zadávat příkazy do zásoby bez povolení superuživatele
--
Pro šifrované zpřístupnění terminálu vzdáleného počítače použijeme
• telnet
• ssh
• rsh
• ftp
• scp
--
Které tři z vyjmenovaných aplikací patří mezi unixové shelly?
• bash, ssh, ksh
• ssh, telnet, putty
• sh, bash, ksh
• bash, sh-bash, ksh-bash
--
Proces běžící na popředí pozastavím speciálním znakem
• intr
• eof
• susp
• supr
• su
--
Nápovědu k příkazu `rm` získám pomocí
• rm -f
• rmman
• rm help
• man rm
--
Proces je v unixu jednoznačně identifikován
• uživatelem, který spustil
• číslem přiřazeným při spuštění
• terminálem, ze kterého byl spuštěn
• časem spuštění
--
Speciální znak **susp**
• přepne počítač do úsporného režimu
• pozastaví proces
• ukončí proces
• zamkne relaci uživatele
--

Příkaz **ps**
• vypíše přihlášené uživatele
• vypíše spuštěné procesy
• vypíše programy v systému souborů
--
Příkaz `kill -9 1000`
• násilně ukončí program s názvem 1000, mám-li na to právo
• násilně ukončí program s názvem 1000
• násilně ukončí proces s číslem 1000, mám-li na to právo
• násilně ukončí proces s číslem 1000
• je chybný příkaz
--
Uživatel **root**
• smí číst všechny soubory, nesmí je však modifikovat
• smí číst všechny soubory a modifikovat jen ty, které jsou ve skupině `root`
• uživateli root se přístupová práva nekontrolují
--
Uživatelů **root** smí být v systému zavedeno
• žádný
• jeden
• dva
• tolik, kolik jich je zavedeno v souboru /etc/passwd
--
Příkaz `man cat`
• vypíše obsah souboru cat na standardní výstup
• vypíše obsah souboru man na standardní výstup
• vypíše manuálovou stránku příkazu cat na standardní výstup
• vypíše manuálovou stránku příkazu man na standardní výstup
• je chybný příkaz
--
Příkaz `man 5 passwd`
• může vypsat manuálovou stránku příkazu passwd
• může vypsat manuálovou stránku souboru /etc/passwd
• může vypsat manuálovou stránku příkazu login
• je chybný příkaz
--
Kterým příkazem **nelze** získat obsah (předpokládejme textového) souboru
• cat
• more
• less
• pwd
• všemi uvedenými
--
Adresář /etc je určen
• pro dočasné soubory jako pomocný
• pro různé soubory, které nebylo možné dát do jiných adresářů
• pro umístění zpravidla konfiguračních souborů
• pro umístění zpravidla speciálních souborů
--
Externí příkazy shellu jsou zpravidla umístěny v adresáři
• /usr/local/bin
• /bin
• /etc
• /usr
• /usr/lib
--
Pro umís»ování adresářů se soubory, jejichž obsah se často mění (např. log soubory se záznamy o provedených operacích), je určen adresář
• /bin
• /lib
• /usr
• /var
• /etc
--
Za základní systém souborů považujeme, ten
• který je nejrozšířenější z podporovaných
• který je na hlavním disku systému
• který je na disku C:
• který obsahuje kořenový adresář systému
--
Mezi základní systémy souborů patří
• iso8859-2, sysfs
• nfs, iso9660
• ext3, xfs
• vfat, tmpfs
--
Mezi typické doplňující systémy souborů patří
• vfat, iso9660, ntfs
• swap, ext3, tmpfs
• ntfs, iso8859, xfs
--
V souboru `/etc/fstab` je
• výstup kontroly systému souborů příkazem fsck
• seznam vadných bloků na zařízeních

• připojení systémů souborů k zařízením
• seznam uživateli dostupných systémů souborů
• takový soubor neexistuje
--
Při neřízeném vypnutí stroje (např. při výpadku napájení) může dojít ke ztrátě dat zapisovaných do systému souborů
• plánovaných zapsat po výpadku
• během výpadku a plánovaných zapsat po výpadku
• před výpadkem, během výpadku a plánovaných zapsat po výpadku
--
Operace spojená s poznačováním obsahu vyrovnávacích pamětí se nazývá typicky
• set
• save
• flush
• write
--
Jméno souboru či adresáře smí být dlouhé nejvýše
• 8+3 znaků
• 16+3 znaků
• 128 znaků
• 255 znaků
--
Maximální délka souboru v unixu je
• 255 znaků
• 256 znaků
• 128 znaků
• je omezena vlastnostmi konkrétního systému souborů
--
Soubory, jejichž jméno začíná tečkou
• jsou přístupné pouze superuživateli
• se nezahrnují do *-expanze
• se nepoužívají
• takové jméno souboru není možné
--
Příkaz `ls -la` **ne**vypíše
• čas změny souboru
• číslo i-uzlu, ve kterém je uložen příslušný soubor
• velikost souboru
• soubory, jejichž jméno začíná tečkou
• adresářové položky . a ..
--
Který příkaz vypíše počet odkazů na jednotlivé soubory?
• ls
• ls -l
• ls -a
• ls -i
--
Relativní cesta k souboru nebo k adresáři začíná vždy
• lomítkem
• běžným adresářem
• domovským adresářem
• znakem ~
--
Rozdíl mezi absolutní a relativní cestou k souboru či adresáři je
• absolutní cesta začíná vždy domovským adresářem
• absolutní cesta začíná vždy běžným adresářem
• absolutní cesta začíná vždy lomítkem
--
K oddělování adresářů a jména souboru v zápisu cesty se používá znak
• \
• /
• |
--
V hierarchii adresářů stojí nejvýše adresář
• /
• C:/
• C:/, D:/, E:/, ...
• root
• /root
--
Běžný adresář jako domovský si uživatel nastaví
• příkazem cd
• editací souboru /etc/passwd
• příkazem pwd
• nenastaví
--
Kterým příkazem se změní běžný adresář vždy na můj domovský adresář?
• cd ~
• cd -
• cd .
• cd /
--
Co se stane po provedení příkazu "`cd –`"?
Bude nastaven

• předchozí pracovní adresář
• domovský adresář aktuálního uživatele
• domovský adresář superuživatele
• speciální adresář definovaný v /etc/passwd
--
Prázdný adresář je adresář
• je adresář s nulovou velikostí
• je adresář s prazdnou tabulkou adresáře
• obsahující dvě konkrétní položky
--
Cestu k běžnému adresáři předá na standardní výstup příkaz
• cd
• pwd
• pcd
• wd
• grep $LOGNAME /etc/passwd|cut -d: -f6
--
Jestliže v adresáři /home/user zadate příkaz `cd ./..` , potom váš běžný adresář bude
• /home
• /
• nelze zadat více argumentů, vypíše se chyba
• /home/user
• domovský adresář
--
Po zadání příkazu `cd /tmp/data` v adresáři /home/user se změní běžný adresář na
• /home/user/tmp/data
• /home/tmp/data
• /tmp/data
• /tmp/data/home/user
--
Pracovní adresář je nastaven na /home/user/data. Jaký bude pracovní adresář po provedení příkazu `cd ./../../.`
• /
• /home
• /home/user
• /home/user/data
--
Vytvoření a zrušení adresáře provedou příkazy
• md a rd
• mdir a rdir
• mkdir a rmdir
• make a remove
--
Nemám-li povoleno v proměnné PATH prohledávání běžného adresáře, spustím proveditelný soubor `muj` příkazem
• run muj
• ../muj
• ./muj
• muj
--
Adresář v systému souborů je uložen v souboru
• adresář není uložen v souboru
• obsahujícím čísla i-uzlu
• obsahujícím jména souborů a čísla i-uzlu
• obsahujícím cestu, jména souborů a čísla i-uzlu
• obsahujícím identifikaci majitele, cestu, jména souborů a čísla i-uzlu
--
Kořenový adresář má číslo i-uzlu
• 0
• 1
• 2
• 3
• -1
--
Kořenový adresář je v UNIXu označen
• /root
• \root
• \
• /
• C:\
--
Položka '.' má v adresáři přiřazené číslo i-uzlu
• 0
• 1
• 2
• shodné s číslem i-uzlu tohoto adresář
• shodné s číslem i-uzlu rodičovského adresáře
--
Pokud je ve výpisu obsahu adresáře u adresáře "." uvedeno číslo i-uzlu 2 a u adresáře ".." 0, potom se jedná o
• běžný (pracovní) adresář
• nesmysl
• adresář druhé úrovně např. /home
• adresář třetí úrovně např. /home/user

--
Příkazem rmdir se typicky maže adresář obsahující
• 0 položek
• 1 položku
• 2 položky
• 3 položky
--
i-uzel
• obsahuje adresu počítače v síti
• obsahuje informace o směrování paketů v uzlech sítě
• obsahuje informaci o jednom souboru či adresáři systému souborů
• obsahuje informaci o jednom uzlu n-uzlového clusteru
--
Zdrojový soubor zruší příkaz
• mv
• cp
• cat
• ls
--
Soubory rušíme příkazem
• mv
• cp
• rm
• cat
• del
--
Systém vždy fyzicky odstraní soubor z tabulky i-uzlů (tzn. soubor se zruší), když
• provedu příkaz `rm`
• počet odkazů na počítadle v i-uzlu klesne na 0
• se uživatel odhlásí, poté co provedl příkaz `rm`
• se vysype koš
• fyzicky zruším i-uzel příkazem rmnode
:r1 příkazem `rm` odstraníte jen jeden z odkazů, další ještě mohou existovat
--
Co se stane po provedení příkazu "`rm -rf *`"?
Bude smazán obsah pracovního adresáře
• vč. jmen začínajících tečkou a vč. podadresářů a bez všech varování
• vč. jmen začínajících tečkou, ale bez podadresářů
• bez jmen začínajících tečkou, vč. podadresářů a bez všech varování
• bez jmen začínajících tečkou, bez podadresářů a bez všech varování
• vč. jmen začínajících tečkou a vč. podadresářů s varováními
--
i-uzel obsahuje
• všechny informace o souboru vyjma dat souboru
• všechny informace o souboru vyjma cesty k souboru a dat
• všechny informace o souboru vyjma jména souboru, cesty k souboru a dat
• všechny informace o souboru vyjma identifikace majitele, jména souboru, cesty k souboru a dat
• všechny informace o tomto uzlu v síti
--
Kořenový adresář má pro položky '.' a '..'
• čísla i-uzlu o jedničku větší
• čísla i-uzlu o jedničku menší
• čísla i-uzlu stejná
• žádná z ostatních odpovědí není správná
--
Pokud příkazem `mv` přesouváte soubory v rámci jednoho systému souborů, pak se číslo i-uzlu přesouvaného souboru
• změní
• nezmění
• zruší
--
Co provede příkaz `ls -a` (označte nejsprávnější odpověď):
• vypíše seznam všech podadresářů
• vypíše seznam všech souborů
• vypíše seznam všech podadresářů a souborů a položek se jménem začínajícím tečkou
• vypíše seznam všech položek se jménem začínajícím tečkou
• provede totéž jako příkaz ls bez parametru
--
Kterým příkazem lze smazat neprázdný adresář?
• rmdir
• rm -r
• rm -a
• rm -f
• neprázdný adresář nelze smazat
--
Který příkaz předá na standardní výstup obsah proměnné PATH?
• show $PATH
• echo $PATH
• echo PATH
• ls $PATH
--
Příkazem `pwd`

• změníme pracovní adresář
• vypíšeme pracovní adresář
• změníme domovský adresář
• vypíšeme domovský adresář
--
Příkazem `rmdir`
• vymažeme prázdný adresář
• přesuneme prázdný nebo neprázdný adresář do koše
• přesuneme prázdný adresář do koše
• vyprázdníme koš
• vymažeme skrytý adresář
--
Maximální počet i-uzlů v systému souborů
• lze dle potřeby měnit příkazem ls -il
• lze dle potřeby měnit příkazem mkfs
• nelze měnit bez nového vytvoření systému souborů
--
Počet odkazů na soubor je uložen
• v adresáři
• v i-uzlu
• v souboru
• není uložen, vždy se při potřebě vypočítává
--
Na soubor vytvořený běžným způsobem textovým editorem
• je nulový počet tvrdých odkazů
• je jeden tvrdý odkaz
• jsou dva tvrdé odkazy
--
Jakým příkazem vytvoříme tvrdý odkaz na soubor?
• ln -t soubor odkaz
• ln -h soubor odkaz
• ln soubor odkaz
• ln -s soubor odkaz
--
Po provedení posloupnosti příkazů
`touch a; ln a b; ln a c`
vzniknou následující počty tvrdých odkazů na jednotlivé soubory
• a 1, b 2, c 3
• a 3, b 2, c 1
• a 2, b 2, c 2
• a 3, b 3, c 3
• příkaz je chybný
--
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
• 2
• 3
• 4
• 5
--
Po provedení příkazu `ln aa bb` přibude v běžném adresáři na výpise příkazu `ls -l` položka
• `lrw-r--r--  1 brandejs staff  6 dub  8 21:50 bb`
• `-rw-r--r--  1 brandejs staff  6 dub  8 21:50 bb`
• `lrw-r--r--  2 brandejs staff  6 dub  8 21:50 bb`
• `-rw-r--r--  2 brandejs staff  6 dub  8 21:50 bb`
--
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?
• odkaz na soubor a
• odkaz na soubor b
• vznikly zároveň
• nelze určit
• výpis je nesmyslný
--
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'?
• výhradně uživatel brandejs
• výhradně uživatel staff
• výhradně člen skupiny staff
• výhradně člen skupiny brandejs
• nelze určit
--
Který příkaz úspěšně skončí po provedení posloupnosti příkazů
`touch a; mkdir b`
• ln a c
• ln b c
• žádný z uvedených příkazů
--

Uživatel xnovak vytvořil příkazem **ln** tvrdý odkaz na soubor, který vlastní uživatel xpolak. Kdo bude ve výpise příkazu `ls -l` uveden jako vlastník odkazu?
• xnovak
• **xpolak**
• oba
• nikdo z nich
--
Kolik bude tvrdých odkazů na adresář a, provedete-li posloupnost příkazů
```
mkdir a; mkdir a/b; mkdir a/c; touch a/d
```
• 2
• 3
• **4**
• 5
• posloupnost příkazů je nesmyslná
--
Kolik bude tvrdých odkazů na adresář a, provedete-li posloupnost příkazů
```
touch a; touch a/b; touch a/c; touch a/d
```
• 2
• 3
• 4
• 5
• **posloupnost příkazů je nesmyslná**
--
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
```
• 2
• **3**
• 4
• 5
• posloupnost příkazů je nesmyslná
--
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
```
• **2**
• 3
• 4
• 5
• posloupnost příkazů je nesmyslná
--
Po provedení posloupnosti příkazů
```
mkdir a; mkdir a/b; ls a; ls -d a
```
se na standardní výstup předají řádky obsahující
• a
  b
• **b**
  a
• a
  a
• b
  b
• posloupnost příkazů je nesmyslná
--
Po provedení posloupnosti příkazů
```
touch a; mkdir b; touch b/a; ls a; ls b
```
se na standardní výstup předají řádky obsahující
• a
  b
• b
  a
• **a**
  a
• b
  b
• posloupnost příkazů je nesmyslná
--
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
• a, c jsou soubory, b je speciální soubor
• **a, c jsou soubory, b je adresář**
• a, b jsou adresáře, c je soubor
• a, c jsou adresáře, b je soubor
• výpis je nesmyslný
--
Tvrdý odkaz se **ne**povolí udělat na
• speciální soubor
• symbolický odkaz
• **adresář**
• soubor
--
Symbolický odkaz se **ne**povolí udělat na
• adresář

• soubor v jiném systému souborů
• kořenový adresář
• speciální soubor /dev/null
• i-uzel
--
Provedením příkazu ln (bez voleb) se počet odkazů na objekt typicky
• zvyšuje
• snižuje
• nemění
--
Provedením příkazu rm se počet odkazů na objekt typicky
• zvyšuje
• snižuje
• nemění
--
Provedením příkazu ln -s se počet odkazů na objekt typicky
• zvyšuje
• snižuje
• nemění
--
Příznak, že objekt je symbolický odkaz, je uveden
• v adresáři
• v i-uzlu
• v souboru
• není uveden nikde
--
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
• a je symbolický odkaz, b je tvrdý odkaz
• c je symbolický odkaz, b je tvrdý odkaz
• b je symbolický odkaz, a je tvrdý odkaz
• b je symbolický odkaz, c je tvrdý odkaz
• výpis je nesmyslný
--
Jaká je velikost dat souboru se symbolickým odkazem ukazujícím na soubor `mujprog.c`
• žádná data se k symbolickému odkazu neukládají
• velikost dat je shodná s velikostí souboru `mujprog.c`
• 0 bajtů
• 9 bajtů
--
Co se předá na standardní výstup po provedení posloupnosti příkazů
`touch a;ln -s a b;rm a;ls`
• a
• b
• posloupnost příkazů je chybná, protože nelze smazat soubor, na který ukazuje symbolický odkaz
--
Dynamickou informaci o uzamknutí souboru (výlučný přístup) obsahuje
• i-uzel na disku
• paměťová kopie i-uzlu
• je uložena mimo i-uzly
--
Informaci o přístupových právech souboru obsahuje
• i-uzel na disku
• výhradně paměťová kopie i-uzlu
• je uložena mimo i-uzly
--
Superblok je
• oblast pro uložení dat souborů
• informační struktura o systému souborů
• náhradní blok za vadné sektory na disku
• blok dat se zašifrovanou informací
--
Speciální soubor je
• totéž co soubor 'skrytý'
• soubor přístupný pouze superuživateli
• zpřístupnění v/v zařízení
• zvláštní soubor jako adresář, symbolický odkaz apod.
• soubor se speciálními právy
--
Speciální soubory jsou ve výpisu příkazu `ls -l` identifikovány v prvním sloupci znakem nebo znaky
• k, l
• s
• x
• b, c
• -
--
Typickým příkladem speciálního souboru je
• /
• /etc/passwd

• /var/mail/xnovak
• /dev/null
• /bin/ls
--
Blokový nebo znakový typicky je
• adresář
• přístupový kód
• systém souborů
• speciální soubor
• superblok
--
Pojmenovaná roura
• je způsob přenášení dat mezi dvěma počítači
• je speciální soubor pro přenos dat mezi dvěma procesy
• je prázdné zařízení, tzv. koš na bity
• připojovací bod pro zpřístupnění připojeného systému souborů
--
Která posloupnost představuje korektní vytvoření a použití pojmenované roury?
• mkdir roura p;cat /dev/null > roura & cat roura; rm roura
• mknod roura p;cat /etc/passwd > roura & cat roura; rm roura
• mknod roura p;cat /etc/passwd > roura & cat roura; rmnod roura
• mknod roura p; cat roura > /etc/passwd & cat /etc/passwd; rmnod roura
• cat roura > /etc/passwd & mknod roura p; cat /etc/passwd; rm roura
--
Speciální soubory pro zpřístupnění V/V zařízení se typicky ukládají do adresáře
• /specf
• /iofiles
• /io
• /dev
• /devio
--
Jakou bude mít velikost soubor 'x' po provedení příkazů
```
echo 'ahoj' > x; cp /dev/null x
```
• nebude existovat
• 0
• 1
• 4
• 5
--
Přístupová práva k objektům systému souborů se určují zvlášť pro
• vlastníka, členy skupiny, ostatní přihlášené uživatele
• uživatele, členy skupiny, ostatní nepřihlášené uživatele
• uživatele, členy skupiny, ostatní přihlášené uživatele
• vlastníka, členy skupiny, ostatní nepřihlášené uživatele
--
Přístupová práva se nastavují v pořadí a s počátečními písmeny zkratky
• owner, group, world
• root, world, execute
• user, root, others
• user, group, others
--
Přístupová práva k souboru se určují zvlášť pro trojici operací
• čtení, zrušení, spuštění
• provedení, zrušení, kopírování
• zrušení, spuštění, editace
• čtení, zápis, provedení
• modifikace, zkrácení, spuštění
--
Přístupová práva k adresáři se definují zvlášť pro trojici operací
• vytváření, rušení, spouštění
• vstup do adresáře, zrušení, vytvoření
• čtení, zapisování, vstup do adresáře
• čtení, vytváření, rušení
• čtení, spouštění, vstup do adresáře
--
Co znamená přístupové právo "x" pro soubor?
• ze souboru lze číst
• do souboru lze zapisovat
• soubor lze spustit
• do souboru lze vstoupit
--
Co znamená přístupové právo "x" pro adresář?
• adresář můžeme vypsat
• do adresáře můžeme zapisovat
• do adresáře je povoleno vstoupit
• adresář je povoleno spustit
--
Přístupová práva k souboru či adresáři jsou uložena
• v jednom bajtu
• ve dvou bajtech
• na devíti bitech
• ve dvanácti bitech
• ve 128 bajtech
--

Vytvořit nový soubor smím v adresáři, na který mám právo alespoň
• čtení a zápisu
• vytváření a vstupu
• zápisu a vstupu
• čtení a vytváření
• čtení, zápisu a vstupu
--
Zrušit soubor v adresáři smím, pokud mám alespoň
• právo zápisu do souboru
• právo zápisu do adresáře a zápisu do souboru
• právo zápisu a vstupu do adresáře
• právo zápisu a vstupu do adresáře, a právo zápisu do souboru
• právo zápisu do souboru a jsem vlastníkem souboru
--
Jaké právo k adresáři musím alespoň mít, pokud chci vytvořit soubor v tomto adresáři?
• -wx
• rw-
• r-x
• -w-
• rwx
--
V adresáři, který je na výpise příkazu `ls -l` označen `drwxrwxrwx`, smím
• rušit libovolné soubory
• rušit soubory, které vlastním
• rušit soubory, ke kterým mám právo zápisu
• nesmím rušit soubory
• rušit soubory, které vlastním a ke kterým mám právo zápisu
--
V adresáři, který je na výpise příkazu `ls -l` označen `drwxrwxrwt`, smím
• rušit libovolné soubory
• rušit soubory, které vlastním
• rušit soubory, ke kterým mám právo zápisu
• nesmím rušit soubory
• rušit soubory, které vlastním a ke kterým mám právo zápisu
--
V adresáři, který je na výpise příkazu `ls -l` označen `d--x--x--x`, smím
• rušit soubor
• vytvořit soubor
• podadresář nastavit jako běžný
• vypsat jeho obsah příkazem `ls`
• žádná jiná odpověď není správná
--
Soubor mohu číst, pokud mám právo alespoň
• čtení souboru
• vstupu do adresáře a čtení souboru
• čtení adresáře, vstupu do adresáře a čtení souboru
• čtení adresáře, vstupu do adresáře a čtení souboru a vstupu do souboru
--
Do souboru mohu zapisovat, pokud mám právo alespoň
• zapisovat do adresáře
• zapisovat do souboru
• vstupovat do adresáře a zapisovat do souboru
• zapisovat a vstupovat do adresáře, zapisovat a číst soubor
--
Soubor (s proveditelným binárním kódem) smím spustit, pokud mám právo alespoň
• na čtení souboru
• čtení adresáře a spuštění souboru
• vstup do adresáře a čtení souboru
• vstup do adresáře a spuštění souboru
• vstup do adresáře, spuštění souboru a čtení souboru
--
Proces, který spustím ze spustitelného souboru obsahujícího proveditelný binární kód a majícího nastavený SUID bit, poběží vždy pod identifikací (uživatelem)
• uživatele, který vlastní soubor
• uživatele, který proces spustil
• superuživatele
--
Proces, který spustím ze spustitelného souboru obsahujícího proveditelný binární kód a **ne**majícího nastavený SUID bit, poběží vždy pod identifikací (uživatelem)
• uživatele, který vlastní soubor
• uživatele, který proces spustil
• superuživatele
--
Který soubor lze spustit tak, že poběží pod identifikací vlastníka souboru (vypsána jsou přístupová práva z příkazu `ls -l`)
• `rwxr-Sr-x`
• `rws--x--x`
• `rwxr-sr-x`
--
Který soubor lze spustit tak, že poběží pod identifikací uživatele, který jej spustil (vypsána jsou přístupová práva z příkazu `ls -l`)
• `-wS--S---`
• `rws--x--x`
• `rwxr-xr-x`

--
Co znamená SUID bit?
• příznak superuživatele v souboru /etc/passwd
• příznak vlastníka adresáře systému souborů
• příznak měnící identifikaci majitele procesu
--
Jsem-li vlastníkem souboru - s jakými přístupovými právy smím soubor spustit (jedná se o spustitelnou binárku; přístupová práva jsou zapsána osmičkově)?
• 0644
• 0766
• 2264
• 7666
--
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)?
• 0644
• 0766
• 2264
• 3656
--
Přístupová práva zapsaná osmičkově 4522 znamenají
• r-s-w--w-
• -wS--x--x
• rw-r--r-t
• rwx-ws-w-
• r--r-x-w-
--
Přístupová práva zapsaná osmičkově 0731 znamenají
• ---rwx-wx
• rwxrw-r--
• rwx-wx--x
• r-xrwx--r
• --r-wxrwx
--
Symbol 't' v přístupových právech znamená
• objekty v adresáři smí rušit jen vlastník objektu nebo vlastník adresáře
• objekty v adresáři smí rušit jen vlastník objektu
• objekty v adresáři vytvářet jen vlastník adresáře
• vlastníkem nově vytvořeného objektu bude vlastník adresáře
--
Tzv. sticky bitem zapínáme následující chování
• soubory v adresáři smí rušit pouze jejich vlastník
• soubory v adresáři smí rušit pouze superuživatel
• soubory v adresáři smí rušit pouze jejich vlastník a vlastník adresáře
• soubory v adresáři smí vytvářet pouze vlastník adresáře
• soubory v adresáři smí vytvářet pouze vlastník adresáře nebo superuživatel
--
Jaká podmínka je postačující k tomu, abych mohl měnit přístupová práva souboru?
• jsem vlastníkem souboru a mám právo zápisu do souboru
• mám právo zápisu do souboru
• mám právo zápisu do souboru a zápisu do adresáře
• jsem vlastníkem souboru
• mám právo zápisu do adresáře
--
Jaká syntaxe příkazu chmod je chybná?
• chmod go+rx
• chmod go=o
• chmod go+X
• chmod u-s
• chmod x+a
--
Kterým příkazem přidáme ke všem položkám běžného adresáře právo zápisu pro skupinu uživatelů? (Ostatní práva neměňte.)
• chmod g+w *
• chmod w+g *
• chmod w=g *
• chmod g=w *
• chmod w+gx *
--
Kterým příkazem přidáme ke všem položkám běžného adresáře právo čtení pro (úplně) všechny uživatele? (Ostatní práva neměňte.)
• chmod r=a *
• chmod a+r *
• chmod go+w *
• chmod r+a *
• chmod ugo+x *
--
Kterým příkazem přidáte všem spustitelným souborům a všem adresářům právo v běžném adresáři právo spuštění/vstupu pro všechny ostatní? (Práva jiných souborů a jiná práva neměňte.)
• chmod o=x *
• chmod o+x *

• chmod o+X *
• chmod g+x *
• nelze udělat
--
Jakým příkazem nastavíme SUID bit?
• chmod u+s
• chmod g+s
• chmod o+s
• chmod x+s
• chmod s+g
--
Jakým příkazem nastavíme přístupová práva ve tvaru `rw-r--r--`
• chmod 640
• chmod ugo=640
• chmod a=rw,go-w
--
Jaká práva bude mít soubor, pokud provedu příkaz `chmod 6755 soubor`
• rwsr-sr-x
• rwsrwsr-x
• rwsrw-rw-
• rwxrwsr-x
--
K čemu slouží příkaz `chmod g=u`
• nastaví stejné práva pro skupinu jako má vlastník
• nastaví stejné práva pro vlastníka jako má skupina
• zruší všechny práva u vlastníka a skupiny
• žádná z uvedených odpovědí
--
Jsem členem více než jedné skupiny. Kdy má smysl, abych použil příkaz **newgrp**
• chci-li vytvořit novou skupinu v systému
• chci-li se dostat k souborům patřícím do jiné skupiny, než do které se chci přihlásit
• chci-li při vytvoření nového souboru použít jinou skupinu, než do které jsem přihlášen
• chci-li požádat administrátora o mé přihlášení do nové skupiny
--
Řídícím operátorem není
• //
• ||
• &&
• ;;
--
Znak, který následuje za \ se
• nepovažuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
• nepovažuje za řídící. Je-li tím znakem "nový řádek", nepokračuje se na dalším řádku.
• považuje za řídící. Je-li tím znakem "nový řádek", pokračuje se na dalším řádku.
• 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
• ' (apostrof)
• $, ` (obrácený apostrof), \ (následuje-li $, `, ", \, nový řádek)
• ' (apostrof), \
• ` (obrácený apostrof)
--
Znaky uzavřené do dvojice uvozovek ("...") ztrácejí svůj řídící význam s výjimkou
• ' (apostrof)
• $, ` (obrácený apostrof), \ (následuje-li $, `, ", \, nový řádek)
• ' (apostrof), \
• ` (obrácený apostrof)
--
Adresář se jménem "Ferda mravenec" (má ve jméně mezeru) nevytvořím příkazem
• mkdir "Ferda mravenec"
• mkdir 'Ferda mravenec'
• mkdir `Ferda mravenec`
• mkdir Ferda\ mravenec
• mkdir Ferda" "mravenec
--
Co se předá na standardní výstup po spuštění příkazu
`echo \'`
• Bad filename
• echo \'
• \'
• "
• '
--
Co se předá na standardní výstup po spustění příkazu echo \'\\\"\'
• '\"'
• '\\\"'
• \\\"\'
• \\\"
--
Příkaz při ukončení vrací ukončovací kód. Jaký ukončovací kód je?
• číselný; 0 znamená úspěšné ukončení, nenulové znamená ukončení s chybou
• číselný; 1 znamená úspěšné ukončení, 0 znamená ukončení s chybou

• číselný; nenulový znamená úspěšné ukončení, 0 znamená ukončení s chybou
• textový; 'ok' znamená úspěšné ukončení, prázdný řetězec znamená ukončení s chybou
• textový; prázdný řetězec znamená úspěšné ukončení, neprázdný řetězec znamená ukončení s chybou
--
Znak \ má následující význam:
• oddělovač adresářů a jména souboru od adresáře v cestě
• zapnutí pokračovacího řádku
• označení control znaku
• zrušení řídicího charakteru následkujícího znaku
:r2 pokračovací řádek zapíná kombinace \ a nový řádek
--
Znak ' (apostrof) má následující význam:
• všechny znaky v řetězci uzavřeném do dvojice '...' ztrácí řídicí význam
• příkaz uzavřený do dvojice '...' se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
• znaky v řetězci uzavřeném do dvojice '...' ztrácí řídicí význam vyjma řídicích znaků $`\
• označuje control znak
--
Znak " má následující význam:
• všechny znaky v řetězci uzavřeném do dvojice "..." ztrácí řídicí význam
• příkaz uzavřený do dvojice "..." se při expanzi příkazového řádku provede a celý řetězec se nahradí obsahem standardního výstupu
• znaky v řetězci uzavřeném do dvojice "..." ztrácí řídicí význam vyjma řídicích znaků $`
• označuje control znak
--
Kolik obrácených lomítek a kolik apostrofů předá na standardní výstup příkaz `echo '\'"'"\\\'`
• 2 a 1
• 2 a 2
• 3 a 2
• 3 a 3
• 4 a 4
--
Který z příkazů nesmaže soubor x?
• rm x
• rm 'x'
• rm "x"
• rm `x`
• rm \x
--
Příkaz
`sort > soubor1 < soubor2`
• čte standardní vstup ze souboru `soubor2` a standardní výstup zapisuje do souboru `soubor1`
• čte standardní vstup ze souboru `soubor1` a standardní výstup zapisuje do souboru `soubor2`
• čte standardní vstup ze souboru `soubor2` a standardní chybový výstup zapisuje do souboru `soubor1`
• čte standardní vstup ze souboru `soubor1` a standardní chybový výstup zapisuje do souboru `soubor2`
--
Operátory přesměrování jsou
• `>&, >, >>, <<-, <>`
• `<<, >&, <->, <&`
• `<, >, >|, <=, >>>`
• `>>, <<, <$, <, >, $>`
--
Který z příkazů nezkopíruje soubor 'a' do souboru 'b'?
• cp a b
• cat a > b
• cat < a > b
• cat alcat > b
• cp < a > b
--
Jaké je korektní a smysluplné přesměrování vstupu?
• ls < adresar
• cd < adresar
• grep Brno < mesta
• ls > adresar
--
Jaké je korektní a smysluplné přesměrování výstupu?
• echo toto se mi libi > soubor
• cd adresar > soubor
• mv a > b
• grep Brno < mesta
--
Soubor 'a' před provedením příkazu `ls > a`
• musí existovat
• nesmí existovat
• může existovat
--

Soubor 'a' před provedením příkazu `sort < a`
• musí existovat
• nesmí existovat
• může existovat
--
Po provedení příkazu
`echo je > je; rm -rf neni; ls je neni > a 2> b`
• bude soubor 'a' větší než soubor 'b'
• bude soubor 'b' větší než soubor 'a'
• budou soubory 'a' a 'b' stejně velké
• soubor 'b' nebude existovat nebo bude prázdný
• soubor 'a' nebude existovat nebo bude prázdný
--
Po provedení příkazu
`echo a > b > c`
• bude soubor 'b' prázdný a soubor 'c' neprázdný
• budou soubory 'b' a 'c' stejně velké
• bude soubor 'b' neprázdný a soubor 'c' prázdný
• soubor 'b' nebude existovat
• soubor 'a' se vyprázdní
--
Po provedení příkazu
`echo b > s; echo a >> s;sort < s > s`
• bude soubor 's' prázdný
• bude soubor 's' obsahovat řádky v pořadí 'a' a 'b'
• bude soubor 's' obsahovat řádky v pořadí 'b' a 'a'
• nebude soubor 's' existovat
:r1 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
• 5
• 6
• 13
• 14
• 15
--
Posloupnost příkazů
`cat <<ukazka`
`ls -l ukazka`
`ukazka`
předá na standardní výstup
• obsah souboru 'ukazka', výstup příkazu 'ls -l ukazka', spustí se program 'ukazka' z běžného adresáře
• text 'ls -l ukazka'
• text 'ukazka'
• nepředá se nic
• předá se výstup programu 'ukazka'
--
Spojení příkazů do kolony je
• ls|grep txt|sort
• ls;grep txt;sort
• ls&grep txt&sort
• ls&&grep txt&&sort
--
Návratový kód seznamu
`rm -rf adr;mkdir adr;touch adr/so;ls adr/so|sort|grep soubor`
bude
• 0
• nenulové kladné číslo
• nenulové záporné číslo
• prázdný řetězec
• text s chybovým hlášením
--
Návratový kód seznamu
`rm -rf adr;touch adr/soubor;echo adr/soubor|sort|grep soubor`
bude
• 0
• nenulové kladné číslo
• nenulové záporné číslo
• prázdný řetězec
• text s chybovým hlášením
--
Příkaz spustíme na pozadí, když
• za příkaz napíšeme &
• před příkaz napíšeme &
• za příkaz napíšeme $
• před příkaz napíšeme $
--
Pro příkaz spuštený na pozadí neplatí
• lze ho ukončit speciálním znakem INTR

• lze ho ukončit příkazem kill
• standardní vstup se napojí na /dev/null
• byl spuštěn pomocí oddělovače &
--
Kam je napojen standardní vstup procesu spuštěného na pozadí?
• /dev/null
• /dev/tty
• /dev/zero
• /dev/console
--
Který z příkazů předá na standardní výstup nejprve řádek obsahující 'b' a potom řádek obsahující 'a'?
• sleep 2;echo a&sleep 1;echo b
• sleep 1;echo a&sleep 2;echo b
• (sleep 1;echo a)&sleep 2;echo b
• (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
• b
• předají se řádky 'a' i 'b' ve stejný čas v náhodném pořadí
• nepředá se nic
--
Provedete posloupnost příkazů/operací:
```
cat > a&cat > b
ahoj
^D
```
Který soubor bude větší?
• a
• b
• soubory 'a' i 'b' budou stejně velké a neprázdné
• soubory 'a' i 'b' budou stejně velké a prázdné
• jedno ^D nestačí, musí se stisknout dvakrát
--
Jakým způsobem nelze úlohu dostat pod správu Job Controllu?
• příkazem fg
• příkazem bg
• spuštěním s &
• speciálním znakem SUSP (^Z)
--
Úloha běžící na pozadí se přesune na popředí příkazem
• lg
• kill
• nohup
• fg
• žádná jiná odpověď není správná
--
Pro seznam neplatí
• je to posloupnost žádneho nebo více příkazu
• příkazy jsou odděleny novým řádkem nebo středníkem (;) nebo ampersandem (&)
• shell provádí příkazy v pořadí, v jakém jsou zapsány
• návratový kód seznamu je návratový kód prvního prováděného procesu
• pokud příkaz končí ampersandem (&), ihned se zahájí provádení následujiciho příkazu
--
Oddělením dvou příkazů oddělovačem &&
• spustíme první příkaz na popředí a druhý příkaz na pozadí
• spustíme první příkaz na pozadí a druhý příkaz na popředí
• provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
• provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód
--
K čemu slouží oddělovač && mezi dvěma příkazy?
• druhý příkaz se provede, pokud je návratový kód prvního nenulový
• druhý příkaz se provede, pokud je návratový kód prvního nula
• druhý příkaz se spustí hned po spuštění prvního
• druhý příkaz se spustí na pozadí hned po spuštění prvního
--
Oddělením dvou příkazů oddělovačem ||
• spustíme první příkaz na popředí a druhý příkaz na pozadí
• spustíme první příkaz na pozadí a druhý příkaz na popředí
• provedeme druhý příkaz, jen když první příkaz vrátil nulový návratový kód
• provedeme druhý příkaz, jen když první příkaz vrátil nenulový návratový kód
--
K čemu slouží oddělovač || mezi příkazy?
• druhý příkaz se provede, pokud je návratový kód prvního nenulový
• druhý příkaz se provede, pokud je návratový kód prvního nula
• druhý příkaz se spustí hned po spuštění prvního
• druhý příkaz se spustí na pozadí hned po spuštění prvního
--
Příkazem
```
ls adresar 2>/dev/null || mkdir adresar
```
• vytvoříme adresář, pokud neexistuje
• vytvoříme adresář, pokud příkaz ls vrátil nulový návratový kód

• vypíšeme vždy obsah adresáře a vytvoříme nový adresář
• vypíšeme obsah adresáře
--
Vyberte nesprávné tvrzení o seznamu
`cd zkusto&&rm *`
• pokud adresář zkusto nebude existovat, zruší se všechny soubory běžného adresáře
• první příkaz bude mít návratovou hodnotu 0, pokud existuje přístupný adresář zkusto
• pokud adresář zkusto bude existovat, zruší se v něm všechny soubory
• pokud první příkaz bude mít návratovou hodnotu 1, tak se druhý neprovede
--
Příkaz
`rm \`cat files.txt\``
• vymaže všechny soubory, které jsou uvedeny v souboru files.txt
• vymaže soubor files.txt
• vymaže soubor cat i soubor files.txt
• vymaže soubor files.txt a také soubory, které jsou v něm uvedeny
--
Příkaz
`rm cat files.txt`
• vymaže všechny soubory, které jsou uvedeny v souboru files.txt
• vymaže soubor files.txt
• vymaže soubor cat i soubor files.txt
• vymaže soubor files.txt a také soubory, které jsou v něm uvedeny
--
Příkaz
`rm –rf *;touch a;echo 'ls' \`ls\``
předá na standardní výstup
• ls a
• a ls
• obsah souboru 'a'
• 'ls' ls
• ls \`ls\`
--
Příkaz
`echo a > seznam;rm 'seznam'`
smaže soubor
• a
• seznam
• příkaz je chybný
• 'a' i 'seznam'
--
Příkaz
`echo a > 'seznam';rm \`cat seznam\``
smaže soubor
• a
• seznam
• příkaz je chybný
• 'a' i 'seznam'
--
Příkaz
`echo a > 'seznam';rm \`echo seznam\``
smaže soubor
• a
• seznam
• příkaz je chybný
• 'a' i 'seznam'
--
Příkaz
`echo a > \`seznam\`;rm 'cat seznam'`
smaže soubor
• a
• seznam
• příkaz je chybný
• 'a' i 'seznam'
--
Proměnná se v shellu "deklaruje" příkazem
• var ADRESAR=/tmp
• ADRESAR=/tmp
• $ADRESAR=/tmp
• strings $ADRESAR /tmp
--
Jak vytvořím proměnou ADRESAR obsahující slovo EMPTY
• ADRESAR=EMPTY
• ADRESAR:=EMPTY
• $ADRESAR=EMPTY
• touch ADRESAR < EMPTY
--
Zadali jsme
`velikost=maxi`
Jak vypíšeme řetězec "maxipes"?
• echo $velikost."pes"
• echo $[velikost]pes
• echo $velikostpes
• echo ${velikost}pes

--
Deklarace proměnné spolu s jejím naplněním se provádí
• jmeno_promenne=[hodnota]
• jmeno_promenne:=[hodnota]
• jmeno_promenne<[hodnota]
• $jmeno_promenne:=[hodnota]
--
Uživatelem zavedené proměnné se v shellu dědí do potomků
• vždy
• nikdy
• jen proměnné označené příkazem export
• jen proměnné označené příkazem import
• jen proměnné označené příkazem child
--
Jaké je nesprávné použití obsahu proměnné ADRESAR?
• echo $ADRESAR/NEW
• echo ${ADRESAR}NEW
• echo $ADRESAR NEW
• echo $ADRESARNEW
--
Co provádí příkaz set spuštěný bez voleb a argumentů?
• vyprázdní všechny proměnné
• smaže všechny proměnné
• vypíše všechny proměnné
• exportuje všechny proměnné
--
Ve kterém způsobu závorkování se provede příkaz ve vnořeném shellu?
• (prikaz)
• { prikaz;}
• <prikaz;>
--
Příkaz
```
cd `echo $OLDPWD`
```
se chová stejně jako:
• pwd
• cd ~
• cd -
• cd ..
--
Který z příkazů má správnou syntaxi?
• { echo ahoj;}
• { echo ahoj }
• {echo ahoj};
• {echo ahoj}
• {echo ahoj }
--
Máme nastavený pracovní adresář /tmp/data. Které z použití příkazu **pwd** vypíše jiný pracovní adresář než /tmp/data?
• cd $PWD;pwd
• { cd $PWD;};pwd
• { cd $HOME;};cd -;pwd
• { cd $HOME;};pwd
• (cd $HOME);pwd
--
Co je uloženo v proměnné PWD
• pracovní adresář
• heslo, zašifrované jednosměrnou šifrou
• domovský adresář
• hesla všech uživatelů (zašifrovaná jednosměrnou šifrou)
--
Mezi složené příkazy nepatří
• for
• case
• while
• until
• find
--
Co vykonává příkaz true?
• vrací hodnotu true
• vypisuje řetězec true dokud není ukončen
• vrací návratový kód 0
• vrací návratový kód 1
• není to příkaz, je to logická hodnota
--
[je
• příkaz
• otevírací závorka pro zápis podmínky
• otevírací závorka pro zápis osmičkového čísla
• symbol přesměrování
• speciální znak
--
Který z příkazů nesmaže všechny soubory z běžného adresáře?
• for S in *; do rm $S; done
• rm `ls`

• rm `echo *`
• for S in `ls`; do rm $S; done
• všechny ostatní příkazy soubory smažou
--
Kterým příkazem pošleme správně celý text e-mailem na zadanou adresu?
• echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík>mail adresa@nekde
• (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)>mail adresa@nekde
• (echo Milý Jeníčku,;echo posílám Ti seznam souborů;ls;echo;echo Tvůj Pepík)|mail adresa@nekde
• 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?
• for JM in $PWD; do echo $JM ${JM}.o; done
• for $JM in *; do echo $JM $JM.o; done
• for JM in *; do echo $JM $JM.o; done
• for JM in *; do echo JM JM.o; done
--
Jaká je korektní syntaxe zápisu vyhodnocení podmínky?
• if `ls soubor`; then rm soubor; fi
• if 'ls soubor'; then rm soubor; fi
• if ls soubor; then rm soubor; fi
• if [ls soubor]; then rm soubor; fi
--
Kterým příkazem **ne**zjistím, zda položka 'adresar' je adresář?
• if test -d adresar; then echo ok; fi
• if [-d adresar]; then echo ok; fi
• if ls -ld adresar|grep -q '^d'; then echo ok; fi
• if pwd adresar; then echo ok; fi
--
Který příkaz je syntakticky špatně?
• if [$# -ge 3] && [$# -lt 6]; then echo ok; fi
• if [$# -ge 3 && $# -lt 6]; then echo ok; fi
• if [$# -ge 3 -a $# -lt 6]; then echo ok; fi
--
Při kterém volání skriptu
`#!/bin/bash<br />if [ $# = 4 -a $1 != $2 ]; then echo ok; fi`
se předá na standardní výstup 'ok'?
• `./skript 'a' `echo a` a A`
• `./skript 1 2 '' 3 4`
• `./skript 1 2 '' 4`
• `./skript a \a b \b`
--
Který soubor nevypíše příkaz
`ls x?[a-c]*`
• žádný soubor, který by měl jméno delší než 4 znaky
• soubor x1.c i když se v adresáři nachází
• soubor xabc i když se v adresáři nachází
• soubor x?abc* i když se v adresáři nachází
--
Jaká je správná syntaxe dotazu na existenci souboru?
• if [-f soubor] then echo ano fi
• if [-f soubor] then; echo ano fi;
• if [-f soubor]; then echo ano; fi
• if [-f soubor;]; then echo; ano; fi
• if [-f soubor;]; then echo ano; fi;
--
Který příkaz je syntakticky správně a současně obsahuje co nejméně mezer?
• [-d adresar]&&echo ano
• [-d adresar]&&echo ano
• [-d adresar]&&echo ano
• [-d adresar] &&echo ano
• [-d adresar] && echo ano
--
Pokud proměnná A obsahuje číslo větší než 5, který příkaz podmínku `A > 5` korektně vyhodnotí a předá na standardní výstup 'ano'?
• if [$A > 5] then echo ano fi;
• if [$A > 5]; then echo ano; fi
• if [$A -gt 5] then echo ano; fi;
• if [$A -gt 5]; then echo ano; fi
--
Kterým/i znakem/znaky začínají v shellu komentáře?
• #
• //
• !
• %
• /*
--
Kterým příkazem spustím skript v běžném shellu?
• /skript

• .skript
• . skript
• ! skript
• žádná jiná odpověď není správná
--
Chci-li spustit skript příkazem `./skript`, musím mít na soubor se skriptem právo minimálně
• spuštění
• čtení a spuštění
• čtení, zápis a spuštění
• čtení
• zápis a spuštění
--
Mějme skript
```
#!/bin/sh
echo $4$1$3$2
```
Co předá na standardní výstup příkaz `./skript a "" b f`
• f a b
• fab
• af b
• afb
• b a f
--
Co předá na standardní výstup příkaz "echo $2", jestliže byl předtím proveden příkaz "`set -- a b c`"?
• `--`
• a
• b
• c
--
Co předá na standardní výstup příkaz
`(exit 1; exit 0); echo $?`
• nic, shell se ukončí
• 0
• 1
• ?
--
Pracovní soubor v adresáři /tmp s unikátním jménem (žádný jiný aktivní proces spuštěný z téhož skriptu nepoužije stejné jméno) vytvořím příkazem
• touch /tmp/pomocny
• touch /tmp/pomocny.$?
• touch /tmp/pomocny.$$
• touch /tmp/pomocny.$%
• touch /tmp/pomocny.$!
--
Příkaz
`ls ~`
předá na standardní výstup položky z
• běžného adresář
• domovského adresáře
• kořenového adresáře
• adresáře /tmp
• 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`
• aa amanda am da
• aa amanda
• amanda
• aa amanda am da AMANDA
• 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
• aa
• amanda am da
• amanda am da AMANDA
• [a-m][!0-9]
--
Skript, který se spustí poté, co se přihlásím do systému, je uložen v souboru
• /home/.profile
• /home/profile
• ~/.profile

• ~/profile
• /etc/login
--
Mám v běžném adresáři soubory .kshrc, dopis.txt, a.out, prog.c. Co předá na standardní výstup příkaz 'echo *' ?
• *
• .kshrc dopis.txt a.out prog.c
• dopis.txt a.out prog.c .kshrc
• .kshrc
• 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 a1 aa1
• aa1 a1
• 1a a a1 aa1
• aa1 a1 1a
--
Který příkaz předá na standardní výstup číslo 4?
• echo 2+2
• echo A:=2+2;
• echo `(2+2)`
• echo $[2+2]
• echo $2+$2
--
Příkaz A=5;B=2;echo C=$A+$B předá na standardní výstup
• 7
• C=7
• C=5+2
• C=A+B
• nic
--
Který skript se spouští vždy při spuštění nového shellu bash?
• ~/.profile
• ~/.bashrc
• /bash/bashrc
• /bin/bash/profile
--
Jakým příkazem předám na standardní výstup hodnotu prvního pozičního parametru?
• echo $1
• cat $1
• $1
• set $1
• cat '$1'
--
Vytvořte adresář zadaný prvním pozičním parametrem. Který příkaz jej nevytvoří?
• if [! -d $1]; then mkdir $1; fi
• test -d $1 || mkdir $1
• [! -d $1] && mkdir $1
• if test ! -d $1; then mkdir $1; fi
• [-d $1] && mkdir $1
--
Jakým příkazem spustíte skript v běžném adresáři a jako první poziční parametr mu předáte řetězec `ahoj`?
• ./skript 1=ahoj
• ./skript $1=ahoj
• $1=ahoj; ./skript
• ./skript;set -- ahoj
• ./skript ahoj
--
Kterým příkazem smažu všechny soubory zadané pozičními parametry (pozičních parametrů může být libovolné množství)?
• `rm $*`
• `rm `$*``
• `rm *`
• `rm $1 $2 $3 $4 ...`
• `rm `*``
--
Jak **ne**spustíme skript, abychom mu jako poziční parametry předali jména položek běžného adresáře začínající písmenem A?
• ./skript `echo A*`
• ./skript 'ls A*'
• ./skript A*
--
Příkaz `echo $$` předá na standardní výstup
• jméno spuštěného skriptu
• číslo procesu shellu
• počet pozičních parametrů předaných skriptu
• všechny poziční parametry předané skriptu
--
Jaká minimální oprávnění jsou nutná pro spuštění shellového skriptu příkazem `bash` <I>*skript*</I>?
• právo spuštění/provedení
• právo čtení

• právo spuštění/provedení a právo čtení
• právo spuštění/provedení a právo zápisu
• právo spuštění/provedení a právo čtení a právo zápisu
--
Jaká je správná syntaxe příkazu pro porovnání numerického obsahu dvou proměnných?
• `[ $A > $B ]`
• **`[ $A -gt $B ]`**
• `[ $A>$B ]`
• `[ $A-gt$B ]`
--
Příkaz `touch knihovna; test -d knihovna` na jinak prázdném adresáři vrátí návratový kód
• prázdný
• 0
• **1**
--
Příkazem `PS1=ferda` změníme
• implicitní podpis v e-mailu
• nastavení běžného adresáře
• **výzvu na terminálu uživatele**
• jméno mailboxu
• primární aktivní proces
--
Příkazovou substituci uzavíráme do
• `'...'`
• `"..."`
• `` `...` ``
• `<...>`
--
Operátorem `2>`
• přesměrujeme standardní výstup
• **přesměrujeme standardní chybový výstup**
• přesměrujeme standardní vstup
• zavřeme standardní výstup
• zrušíme chybový výstup
--
Kterým příkazem vypíšeme na standardní výstup jména všech adresářových položek, které kdekoli ve svém jméně mají písmeno A?
• echo .*A.*
• ls '.*A*'
• **echo *A***
• ls ".A.*"
• ls *A.*
--
Jakým příkazem spojíme za sebe obsah souborů `a` a `b` a výsledek uložíme do souboru `c`?
• cat < a < b > c
• cat a b | c
• **cat a b > c**
• cat < a < b | c
• a b | cat > c
--
Vytvořte adresář `Mail` v domovském adresáři uživatele, pokud tam již není. Jaký příkaz toto **ne**provede?
• `[ -d ~/Mail ] || mkdir ~/Mail`
• `(cd; test -d Mail || mkdir Mail)`
• `if [ ! -d $HOME/Mail ]; then mkdir $HOME/Mail; fi`
• `(cd ~; mkdir Mail 2>/dev/null)`
• **`test -d ~/Mail && mkdir Mail 2>/dev/null`**
--
Jakým příkazem zkopírujete obsah souboru `a` do souborů `b`, `c`, `d` současně?
• `cat a > b > c > d`
• `cp a b c d`
• `tee < a > b > c > d`
• **`cat a | tee b c > d`**
• `tee a | cat b c d`
--
Jakým příkazem přidáte za konec souboru `x` řádek obsahující `xxx-konec-xxx`?
• **`echo xxx-konec-xxx >> x`**
• `cat x < xxx-konec-xxx`
• `cat x|echo xxx-konec-xxx > x`
• `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.
• `for S in *.txt; do echo $S; done`
• `cat *.txt`
• **`echo *.txt`**
• `ls -l *.txt`
--
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`
• řešení: `1,$s/^[A-Za-z]*[ ][A-Za-z]*[ <][a-z]*[>]/Jmeno: \1 Prijmeni: \3 e-mail: \5/`
• řešení: `1,$s/\(^[A-Za-z]*\)[ ]\([A-Za-z]*\)[ ]\(<[a-z]*>\)/Jmeno: \1 Prijmeni: \2 e-mail: \3/`
• řešení: `1,$s/\(.*\) \(.*\) <\(.*\)>/Jmeno: \1 Prijmeni: \2 e-mail: \3/`
• řešení: `1,$s/^[A-Za-z]*[ ][A-Za-z]*[ <][a-z]*[>]/Jmeno: $1 Prijmeni: $3 e-mail: $5/`
--
V textovém souboru jsou řádky obsahující takto uspořádané informace:
`brandejs:*:11000:100:Michal Brandejs:/home/brandejs:/bin/ksh`
Jakým substitučním příkazem editoru **vi** je všechny hromadně převedete do tvaru:
`Michal Brandejs <brandejs>`
• řešení: `1,$s/(.*):.*:.*:.*:(.*):.*/\2 <\1>/`
• řešení: `1,$s/\(.*\):.*:.*:.*:\(.*\):.*:.*/\2 <\1>/`
• řešení: `1,$s/\([^:]*:\).*:\([^:]:\).*/\2 <\1>/`
--
Regulární výraz [0-9][^0-9].x vyhovuje např. řetězci
• 0ax
• a0x
• 00bx
• 0abx
--
Jakým substitučním příkazem editoru **vi** nahradíme řetězec `\end{syntax}` řetězcem `\end{syntax}%` (stačí provést 1x na řádku)?
• `1,$s/\end\{syntax\}/\end\{syntax\}%/`
• `1,$s/\end{syntax}/\end{syntax}%/`
• `1,$s/\\end{syntax}/\\end{syntax}%/`
• `1,$s/\\\end{syntax}/\\\end{syntax}%/`
--
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?
• `s/=[ ][ ]*//g`
• `s/=[ ]*//g`
• `s/=[ ][ ][ ]*//g`
• `s/=*//g`
• `s/=*[ ]//g`
--
Kterým substitučním příkazem editoru **vi** v celém souboru vyprázdníme všechny řádky, které obsahují pouze řetězec `ahoj`?
• `1,$s/ahoj//g`
• `1,$s/*ahoj*//g`
• `1,$s/[^ ]*ahoj[^ ]*//`
• `1,$s/^ahoj$//`
• `1,$s/\^ahoj\$//`
--
Jakým substitučním příkazem editoru **vi** zrušíme nadbytečné mezery na běžném řádku (tzn. řetězce mezer delší než 1 znak nahradit jednou mezerou)?
• `s/[ ]*/ /`
• `s/ [ ]*/[ ]/`
• `s/[ ][ ]*/ /`
• `s/[ ][ ]*/[ ]/`
--
Jakým substitučním příkazem editoru **vi** přidáme za poslední znak každého řádku znak středník?
• 1,$s/.*/;/
• 1,$s/.*$/;/
• 1,$s/$/;/
• 1,$s/^.*/;/
• 1,$s/$*/;/
--
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,$s/index\.html\{0,1\}$//`
• `1,$s/[a-z]*$//`
• `1,$s/\/index.html*//`
• `1,$s/\/[a-z].[a-z]$//`
--
Jakým substitučním příkazem řádkového režimu editoru **vi** změníte všechny řádky souboru ze tvaru
`Lampa stolní typ GY34827|B240|3487636|790`
do tvaru
`Lampa stolní typ GY34827|B240|DKP 3487636|790 Kč`
• `1,$s/\(.*\)|\(.*\)|\(.*\)|\(.*\)/\1DKP \2 Kč/`
• `1,$s/\(.*|.*|\)\(.*|.*\)/\1DKP \2 Kč/`
• `1,$s/\(.*\)|\(.*\)|[0-9]*|\(.*\)/\1DKP \2 Kč/`

--
Jakým příkazem z proměnné PATH vyřadíme adresář /bin ? Pro jednoduchost předpokládejte, že není ani na začátku, ani na konci.
• **PATH=`echo $PATH|sed 's/:\/bin:/:/'`**
• PATH=echo $PATH|`sed 's/:\/bin:/:/'`
• PATH=`sed 's/:\/bin:/:/'`
--
Zda soubor `file` obsahuje textový řetězec `string` zjistíme příkazem
• find string file
• **fgrep string file**
• sed string file
• string string file
• search string file
--
Počet adresářových položek v běžném adresáři zjistíme příkazem
• **ls|wc -l**
• ls -l
• count `ls`
--
Soubory obsahující v názvu podřetězec '_delete' v běžném adresáři a všech jeho podadresářích smažeme příkazem
• for S in *_delete*; do rm $S; done
• **find . -name '*_delete*' -exec rm {} \;**
• rm *_delete*
• for SO in `ls *_delete*`; do rm $SO; done
• for SO in `ls _delete`; do rm $SO; done
--
Který příkaz ze všech souborů v běžném adresáři vyřadí řádky, které v prvním sloupci začínají znakem '#'?
• grep -v \^\# *
• for S in *; do grep -v \# $S > $S; done
• **for S in *; do grep -v '^#' $S > $S.s; mv -f $S.s $S; done**
• find ~ -type f -exec grep -v \^\# { } | dd of={ } \;
--
Kterým příkazem smažeme (pouze) v běžném adresáři soubory obsahující ve svém jméně podřetězec 'tempor'
• for S in *; do grep -q tempor $S && rm $S; done
• find . -name '*tempor*' -exec rm { } \;
• find . -type f -exec grep -q tempor { } \; -exec rm { } \;
• **rm *tempor***
--
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.
• **find ~ -type f \(-name '*brno*' -or -exec grep -q brno {} \; \) -print**
• find ~ -type f -exec grep -q brno { } \; -name '*brno*' -print
• find ~ -type f (-name brno -or -exec grep -q '*brno*' { } \;) -print
• find ~ -type f \(-name '*brno*' -or -exec grep -q '*brno*' { } \; \) -print
--
Kterým příkazem předáme obsah souboru `so` na standardní výstup?
• echo so
• **cat so**
• grep so
• print so
--
Jakým příkazem předáme na standardní výstup řetězec `ahoj`?
• **echo ahoj**
• cat ahoj
• grep ahoj
• cat '*ahoj*'
• echo [ahoj]
--
Jakým příkazem předáme na standardní výstup počet adresářových položek běžného adresáře zvýšený o 2?
• cat $(`ls|wc -`+2)
• $((`ls|wc -`+2))
• wc -l $((`ls`+2))
• **echo $[`ls|wc -l`+2]**
• for SO in *; do echo 2;done|wc -l
--
Jakým příkazem zrušíme všechny podadresáře v běžném adresáři, které obsahují více než 10 položek?
• **for S in *; do [-d $S -a `find $S -print|wc -l` -gt 10] && rm -rf $S; done**
• find . -type d -exec [ls { }|wc -l] > 10 \; -exec rm -rf { } \;
• find . -type d -exec for S in { }; do [ls -R|wc -l -gt 10]; done \; && rm -rf { } \;
• for S in *; do test -d $S && test wc -l `ls -R` -gt 10 && rm -rf $S; done
--
Jakým příkazem přejmenujeme všechny soubory v běžném adresáři mající příponu `.txt` na příponu `.tex`?
• mv *.txt *.tex
• **for S in *.txt; do N=`echo $S|sed 's/\.txt$/.tex/'`; mv $S $N; done**
• find . -type f -name '*.txt' -exec mv { }.txt { }.tex \;
• find . -type f -name '*.txt' -exec mv { } ${%%.txt}.tex \;

• rm *.txt *.tex
--
Do souboru `b` zkopírujte ty řádky ze souboru `a`, které obsahují symbol `>` (větší než).
Jakým příkazem provedete toto zadání?
• cat alsed s/>// > b
• grep > a '>' b
• **grep '>' a > b**
• sed s/>// < a > b
--
Jakým příkazem předáte na standardní výstup jména všech souborů běžného adresáře,
které ve svém těle obsahují textový řetězec `Brno`?
• `ls *Brno*`
• `grep *Brno* *`
• **`grep Brno *`**
• `find . -type f -exec grep *Brno* {} /dev/null \;`
• `echo *Brno*`
--
Jakým příkazem **ne**porovnáme např. shodu dvou řetězců, chceme-li je zadat jako
argumenty na příkazovém řádku?
• `[`
• **`cmp`**
• `test`