Další Předchozí Obsah

5. Přihlášení na vzdálený stroj

5.1 telnet

Kdysi kdosi prohásil, že `telnet` je ta neskvělejší věc, jaká kdy byla na počítačích k vidění. Možnost vzdáleně se přihlásit a pracovat na jiném počítači je to, co odlišuje Unix a Unix-like operační systémy od ostatních OS.

`telnet` vám umožňuje přihlásit se do vzdáleného počítače, jako byste seděli u jeho terminálu. Jakmile je ověřeno vaše uživatelské jméno a heslo, dostanete shell prompt a můžete dělat cokoliv, co na textové konzoli dělat jde.

K přihlášení do vzdáleného stroje použije následující syntaxi:

```
$ telnet stroj [číslo_portu]
```

Číslo portu je nepovinné. Pokud host odpovídá, obržíte výzvu k přihlášení. Zadejte vaše uživatelské jméno a heslo. A je to. Jste v shellu. K ukončení vaší telnet-ové session použije buď příkaz `exit`, nebo příkaz `logout`.

5.2 rlogin

Pro zalogování v lokální síti na jiném uzlu slouží pro stejného uživatele přikaz `rlogin`. Syntaxe je podobná jako u telnetu:

```
$ rlogin [-l uživatel] stroj
```

Je-li lokální uživatel v souboru `.rhosts` vzdáleného počítače, potom se rlogin neptá na heslo.

5.3 rsh

Příkaz `rsh` dává možnost zadat příkaz shellu na jiném uzlu v lokální síti. Seznam stanic, na nichž lze příkaz `rsh` použít, je uveden v souboru `.rhosts`. Oproti příkazům `telnet` nebo `rlogin`, které otevírají seanci na vzdáleném uzlu, příkaz `rsh` neprovádí login, ale pouze provede příkaz. To může být urychlením práce, chceme-li pouze zadat jednotlivý příkaz na vzdáleném uzlu.

```
$ rsh stroj příkaz
```

rsh spustí *remote shell*, který vykoná zadaný příkaz. Na vzdáleném uzlu nelze pracovat interaktivně. Výstup příkazu je směrován na standardní výstup lokálního počítače. Lze rovněž použít operátory pro přesměrování standardního vstupu a výstupu (na lokálním počítači).

V následujícím příkladě je soubor zkomprimován a archív přenesen na jiný uzel, kde je páskové zařízení:

```
$ tar cf - . |rsh sun4 dd of=/dev/nrst0
```


5.4 rcp

Program `rcp` umožňuje kopírování souborů mezi různými uzly v síti. Syntaxe je následující:

```
$ rcp [volby] uživatel@uzel:kopírovaný_soubor uživatel@uzel:kopírovaný_soubor
```

V argumentu volby může být parametr `-r`, který způsobí zkopírování adresáře včetně jeho podadresářů. Například příkaz

```
$ rcp -r report jane@hots2:report
```

zkopíruje adresář `report` včetně podadresářů z lokálního uzlu na uzel `host2` uživateli `jane` do adresáře `report`.

5.5 ftp

Program FTP umožňuje přenos souborů mezi lokálním a vzdáleným uzlem. Uživatel musí mít zavedeno své uživatelské jméno na vzdáleném uzlu. Spuštění programu `ftp` má formát:

```
$ ftp [jméno_stroje]
```

Po zadání příkazu se program spojí se zadaným systémem a bude po vás vyžadovat zadání uživatelského jména a hesla.

Ve světě existuje také velké množství tzv. anonymních serverů. V tom případě se zadává jméno `anonymous` a heslem je vaše e-mailová adresa. Ta samozřejmě může být i falešná.

Když vám program vypíše `ftp>`, můžete začít zadávat příkazy:

`!příkaz`	Opustí prostředí `ftp`. a vykoná lokálně příkaz.
`dir`	Výpis obsahu zdáleného adresáře. Příkaz může mít dva parametry. Prvním můžeme zadat pomocí hvězdičkové konvence, které soubory chceme vypsat. Druhým parametrem je jméno souboru pro uložení výpisu obsahu adresáře.
`lls`	Výpis obsahu lokálního adresáře.
`cd adresář`	Pohyb v adresářovém stromu na vzdáleném počítači.
`lcd adresář`	Pohyb v lokálním adresářovém stromu.
pwd	Jméno aktuálního adresáře na vzdáleném stroji.
`rmdir adresář`	Smaže adresář na vzdáleném stroji.
`mkdir adresář`	Vytvoří adresář na vzdáleném stroji.
`get soubor`	Přečte soubor ze vzdáleného systému.
`put soubor`	Pošle soubor na vzdálený systém.
`mget soubory`	Přenos více souborů najednou.
`mput soubory`	Přenos více souborů najednou.
`ascii`	Nastaví režim přenosu na textové soubory.
`binary`	Nastaví režim přenosu na binární soubory.
`prompt on/off`	Ptá/neptá se na potvrzení každé operace.

`hash`	Při přenosu každého kilobajtu vypíše znak #.
`open stroj`	Naváže spojení se strojem.
`close`	Ukončí spojení.
`quit`	ukončí program.

5.6 Povolení snadného vzdáleného přístupu k vašemu účtu - soubor .rhosts

Jestliže se ve vašem domovském adresáři nachází soubor `.rhosts`, pak lze ze vzdálených počítačů spouštět aplikace na vašem počítači. Uveďme si příklad. Předpokládejme, že jste přihlášeni na počítači `cs.oberlin.edu`. Na počítači `floss.life.uiuc.edu` je správně konfigurován soubor `.rhosts`.

Pak ze svého počítače můžete na počítači `floss.life.uiuc.edu` spustit aplikaci, jejíž výstup bude přesměrován na vaši obrazovku, a dokonce se před tím nebudete muset přihlašovat a zadávat heslo.

Soubor `.rhosts` může vypadat takto:

```
frobnozz.cs.knowledge.edu jsmith
aphrodite.classics.havhaahd.edu wphilps
frobbo.hoola.com trixie
```

Formát souboru je velmi jednoduchý: jméno počítače následované jménem uživatele. Předpokládejme nyní, že uvedený příklad je mým skutečným souborem `.rhosts` na vzdáleném počítači `floss.life.uiuc.edu`. To by znamenalo, že bych mohl spustit program na počítači `floss` a výstup by mohl být přesměrován na kterýkoli počítač uvedený v tomto souboru, pokud bych byl přihlášen jako odpovídající uživatel.

Přesný mechanismus, prostřednictvím kterého uživatel uskutečňuje spouštění vzdáleného programu, je realizován programem `rsh`, jehož název je zkratkou pro *remote shell*, tedy *vzdálený příkazový procesor*. Ten nastartuje příkazový procesor na vzdáleném počítači a spustí specifikovaný program. Uveďme si příklad:

```
frobbo$ whoami
trixie
frobbo$ rsh floss.life.uiuc.edu "ls ~"
foo.txt mbox url.ps snax.txt
frobbo$ rsh floss.life.uiuc.edu "more ~/snax.txt"
[nyní se zobrazí stránky souboru snax.txt]
```

Uživatel `trixie` na počítači `floss.life.uiuc.edu`, který má soubor `.rhosts` uvedený v předcházejícím příkladě, umožňuje uživateli `trixie` na počítači `frobbo.hoola.com` spouštět programy z počítače `floss`.

Soubor `.rhosts` bude pracovat správně, i když nemáte na všech počítačích stejné uživatelské jméno. Chcete-li sdělit vzdálenému počítači, jaké jméno uživatele chcete použít k přihlášení se, použijte při zadání příkazu `rsh` volbu `-l`. Pokud takový uživatel na vzdáleném počítači existuje a pokud zde existuje soubor `.rhosts` obsahující jméno vašeho (t.j. lokálního) počítače a jméno uživatele, pak se příkaz `rsh` provede úspěšně.

```
frobbo$ whoami
trixie
frobbo$ rsh -l larry floss.life.uiuc.edu "ls ~"
[zde se objeví seznam souborů mého adresáře na počítači floss]
```


V souboru `.rhosts` mohou být uvedeny i jiné kombinace - například jméno uživatele následující za jménem vzdáleného počítače se může vynechat a tak umožnit kterémukoli uživateli na vzdáleném počítači spouštět programy na vašem počítači. To je však spojeno s jistými riziky. Nepovolaná osoba by mohla například zrušit vaše soubory. Pokud se rozhodnete pro takto organizovaný soubor `.rhosts`, pak byste se měli ujistit, že právo číst soubor `.rhosts` máte pouze vy.

5.7 Soubor .netrc

V souboru `.netrc` můžete mít uložena některá implicitní nastavení pro `ftp`. Následující řádky obsahují příklad takového souboru:

```
machine floss.life.uiuc.edu login larry password fishSticks
machine darwin.life.uiuc.edu login larry password fishSticks
machine geta.life.uiuc.edu login larry password fishSticks
machine phylo.life.uiuc.edu login larry password fishSticks
machine ninja.life.uiuc.edu login larry password fishSticks
machine indy.life.uiuc.edu login larry password fishSticks
machine clone.mcs.anl.gov login fogel password doorm@
machine osprey.mcs.anl.gov login fogel password doorm@
machine tern.mcs.anl.gov login fogel password doorm@
machine altair.mcs.anl.gov login fogel password doorm@
machine dalek.mcs.anl.gov login fogel password doorm@
machine juju.mcs.anl.gov login fogel password doorm@
machine sunsite.unc.edu login anonymous password
larry@cs.oberlin.edu
```

Každý řádek souboru `.netrc` specifikuje jméno počítače, přihlašovací jméno, které se má použít jako implicitní pro tento počítač, a heslo. Uvedené nastavení vám ušetří velké množství času, protože jinak byste při přihlašování se k jednotlivým serverům ftp museli pokaždé zadávat dlouhé identifikační řetězce. Pokud se budete přihlašovat k serveru `ftp`, jehož jméno je uvedeno v souboru `.netrc`, pokusí se program ftp aplikovat uživatelské jméno a heslo z tohoto souboru.

Při spouštění programu `ftp` můžete pomocí parametru `-n` specifikovat, že nechcete použít implicitní nastavení ze souboru `.netrc`. Příkaz pak bude mít tvar `ftp -n`.

Musíte se ujistit, že soubor `.netrc` jste schopni číst pouze vy. K nastavení příslušných přístupových práv použijte program chmod. Kdyby soubor `.netrc` mohli číst jiní uživatelé, pak by mohli odhalit vaše hesla platná pro servery ftp uvedené v tomto souboru.

5.8 Systém X-Window

X Window Systém (nebo X Window, nebo jenom X) vznikl v Massachusetts Institute of Technology (MIT) a lze jej definovat jako implementaci GUI. X Window Systém je také chráněná značka, kterou drží MIT. Základní práce na X Window Systému probíhaly v letech 1984 až 1987. Z roku 1987 máme verzi X11. Ta se samozřejmě stále inovuje. V současné době se nejvíce používá Release 6.

Zvláštním rysem X Window Systému je to, že správa oken byla vyvinuta jako samostatný program (či soubor programů) a není tedy součástí jádra, jak je tomu zvykem u jiných GUI. Z toho plynou dvě důležité vlastnosti: systém není bezprostředně závislý na tom operačním systému, na kterém byl implementován, a je možné jej používat distribuovaně (v počítačové síti), tzn. že aplikace mohou běžet na jiném počítači, než na kterém se zobrazují.

Abychom si mohli vysvětlit, jak je to zařízeno, musíme se zmínit o základních pojmech: *displej* je technické zařízení, na kterém aplikace zobrazuje své okno, z jehož klávesnice čte příkazy a jehož myší (polohovacím zařízením) ovládá kurzor. Jeden displej může mít více *obrazovek*. Program, který řídí činnost displeje, se nazývá server. Aby nedocházelo k nejednoznačnostem ve výkladu pojmu server,

používá se zde raději označení *X server*. Aplikaci, která někde běží a na některém displeji zobrazuje okno, nazýváme *X klientem*.

Displejem může být konzola pracovní stanice, na které běží jak X server, tak i X klienti. Displejem může také být konzola PC, na kterém běží některý z operačních systémů typu UNIX s implementací X Window Systému. Vyrábějí se též specializovaná zařízení nazývaná *X terminály*. Jsou to vlastně počítače bez disku, mající rychlý procesor, několik MB paměti, klávesnici, myš a zpravidla velkou obrazovku. V procesoru takového zařízení běží pouze X server, X klienty zde není možné spustit. Program X serveru se do X terminálu zpravidla zavádí po síti v okamžiku, kdy X terminál zapínáme.

5.9 Spouštění X klientů

X klienty pod operačním systémem UNIX spouštíme stejně jako jiné programy. Na příkazovém řádku shellu zadáváme vedle jména souboru s X klientem také volby (přepínače; jejich klíčové slovo je uvozeno znakem mínus). Základní sada voleb je pro X klienty standardizována a můžeme je tedy použít u většiny X klientů. Uveďme si ty nejpoužívanější.

Určení displeje a obrazovky

Displej a obrazovka, na které se má X klient zobrazovat, se určuje volbou s touto syntaxí:

`-display` [*počítač*]`:`*displej*[`.`*obrazovka*]

- [*počítač*] je adresa počítače, na kterém běží X server. Pokud adresu neuvedeme (nebo uvedeme `unix`), zobrazí se okno na tom počítači, na kterém běží X klient.
- [*displej*] je číslo X serveru, který spravuje požadovaný displej. Displeje se číslují od 0 a právě hodnota 0 se na tomto místě nejčastěji používá. Parametr *displej* nelze vynechat.
- [*obrazovka*] je číslo obrazovky v rámci zadaného displeje. Nejčastější a implicitní hodnota je 0. Parametr lze tedy vynechat.

Pokud při startu X klienta neuvedeme volbu `-display`, bude se hledat proměnná prostředí `DISPLAY`. Existuje-li, použije se řetězec s ní spojený tak, jako by byl uveden za `-display`.

X Window Systém disponuje prostředky ochrany, které dovolují zobrazovat okna pouze na tom displeji, jehož uživatel to explicitně povolil. Uživatel displeje příkazem `xhost` sdělí lokálnímu X serveru adresy počítačů, ze kterých mohou X klienti displej používat.

Adresy počítačů, které mohou lokální displej používat, načítá X server při startu ze souboru `/etc/X0.hosts` (číslice udává pořadí X serveru). Uživatel potom už za běhu X serveru tento seznam modifikuje následujícími parametry příkazu `xhost`:

[+]*adresa*počítači povolí zobrazovat
-*adresa*počítači zakáže zobrazovat
+povolí zobrazovat všem
-zakáže všem vyjma `X0.hosts`

Uveďme příklad spouštění X klienta `xterm`, který se má zobrazit na počítači `atys.fi.muni.cz`:

```
$ xterm -display atys.fi.muni.cz:0 &
```


A chceme-li všechny X klienty (u nichž explicitně neuvedeme jinou možnost) zobrazovat na počítači `atys`, můžeme situaci elegantně vyřešit nastavením proměnné prostředí (v interpretech příkazů odvozených od `sh`).

Musíme se normálně přihlásit na vzdálený počítač pomocí programu telnet, rlogin nebo ssh. Potom je potřeba spustit X klienta, kterému je ovšem nutné oznámit, se kterým X serverem má spolupracovat. To je možné provést nastavením proměnné prostředí DISPLAY, v Bourne shellu příkazem

```
DISPLAY=server:0; export DISPLAY
```

v Korn shellu a v bash je možné použít i zkrácený zápis

```
export DISPLAY=server:0
```

v C-shellu pak

```
setenv DISPLAY server:0
```

Určení geometrie okna

Další volbou nastavujeme velikost prvního okna X klienta a případně i umístění okna v kořenovém okně. Volba má tuto syntaxi:

`-geometry` *šířka***x***výška*[+-*x*+-*y*]

Údaje jsou dány buď v počtech sloupců a řádků (u textových aplikací, např. `xterm`), nebo v pixelech. Povinné údaje udávají velikost okna. Kladné hodnoty nepovinných udávají umístění okna vzhledem k levému hornímu rohu kořenového okna, záporné hodnoty vzhledem k pravému nebo dolnímu okraji kořenového okna.

Určení fontu

Fontům se v tomto čísle Zpravodaje věnuje článek Libora Škarvady. Na tomto místě si řekněme, že jméno fontu představuje jak jeho řez (tvar), tak i jeho velikost. Jméno fontu uvádíme za volbou `-fn` nebo `-font`. Uveďme si příklad určení fontu pro okno s emulátorem terminálu:

```
$ xterm -fn 10x20 &
```

Emulátor `xterm` používá fonty pevné šířky různých velikostí. Určením fontu nepřímo specifikujeme velikost okna emulátoru. Jeden X klient může používat více fontů, touto volbou však specifikujeme pouze ten základní.

Určení barev

Další typickou vlastností je možnost nastavit barvu popředí, pozadí a rámečku lemujícího okno X klienta. Barva popředí se zadává za volbou `-fg` (nebo `-foreground`), barva pozadí volbou `-bg` (`-background`) a barva rámečku `-bd` (`-border`). Barevné displeje zpravidla ovládají X servery zobrazující 256 možných barev. Z těchto 256 barev bývá 216 barev předurčeno ve formě univerzální palety společné všem X klientům (216 barev vznikne kombinací 6 úrovní od každé barevné složky) a 40 barev zbývá pro volné použití klienty.

Nezávisle na tom, kolik barev X server zobrazuje, určujeme barvu buď jménem (výberem z položek

souboru `/usr/X11/lib/X11/rgb.txt`), nebo kombinací čísel. V tomto druhém případě barvu popíšeme ve tvaru:

`rgb:`*červená*`/`*zelená*`/`*modrá*

Každá barevná složka se zadává šestnáctkovým číslem v intervalu `0` až `ffff`. Např. kombinace `rgb:ffff/ffff/0` představuje žlutou barvu. Tyto údaje se přímo předají X serveru a předpokládá se, že jsou již po gamma korekci (přepočet odpovídající lidskému vnímání barev). Spojitý barevný prostor lze definovat zápisem:

`rgbi:`*červená*`/`*zelená*`/`*modrá*

Zde se zadávají čísla v intervalu 0.0 až 1.0 a chápou se jako intenzity jednotlivých barevných složek. Tyto údaje ještě projdou gamma korekcí. Oba uvedené numerické způsoby zápisu adres byly zavedeny v X11R5. Díky kompatibilitě se starší verzí můžeme ještě použít jiný šestnáctkový zápis:

#*rgb*
#*rrggbb*
#*rrrgggbbb*
#*rrrrggggbbbb*

Ve všech variantách, kde je méně než 16 bitů pro každou barevnou složku, představují uvedené číslice vyšší řády. Tedy #3a7 je totéž co #3000a0007000.

Ikona místo okna

Dává-li uživatel přednost prvotnímu zobrazení klienta ve formě ikony před vykreslením okna, může na příkazový řádek zapsat volbu `-iconic`.

Změna názvu a titulku

Za volbou `-name` uvádíme nový název X klienta. Implicitně je název shodný se jménem souboru, ze kterého byl spuštěn. Nový název se objeví např. na výpise aktivních procesů a jeho pomocí dokážeme konkrétního klienta identifikovat.

Titulek, který většina klientů vypisuje v hlavním okně aplikace, můžeme nastavit či změnit řetězcem uvedeným za volbou `-title`. Řetězec obsahující alespoň jednu mezeru musí být uzavřen do uvozovek.

5.10 Atributy X klientů

Upravovat chování a vzhled X klientů pomocí voleb na příkazovém řádku je sice pohodlné, ale uživateli to poskytuje příliš málo možností.

V originální anglické literatuře se na tomto místě setkáváme s pojmem *resource*, v českých překladech se používá pojem *atribut*. Téměř každá komponenta X klienta se může nastavovat prostřednictvím *atributů*. Atribut se skládá ze jména a hodnoty. Hodnotou je booleovská konstanta, číslo nebo řetězec. Jméno atributu je tvořeno hierarchicky a zahrnuje jak jméno X klienta (předpokládá se, že byl vybudován prostředky knihovny Xt - X Toolkit, či některé z knihoven nadřazených), tak i přesné vymezení komponenty, na kterou se má hodnota atributu aplikovat.

Každý X klient je sestaven z předem přichystaných *přípravků* (v anglických originálech *widgets*). Přípravkem je např. tlačítko, lišta, rolovátko, seznam atd. Přípravky lze do sebe vnořovat. Toto všechno musí syntax jména atributu postihnout. Obecný formát zápisu atributu je následující:

objekt.subobjekt[.subobjekt...].atribut: hodnota

Položkou *objekt* rozumíme buď jméno programu X klienta, nebo konkrétní spuštěnou instanci (podle volby `-name`, bude rozvedeno dále). Položky *subobjekt* korespondují s hierarchií použitých přípravků. Položka *atribut* jednoznačně popisuje vlastnost posledního subobjektu a konečně *hodnota* je to, co se atributu přiřadí. Uveďme příklad:

```
xterm.vt100.scrollBar: True
```

Takto zapsaný atribut zapne (booleovskou hodnotou True) zobrazování posuvné lišty v aplikaci *xterm* v okně VT100. Konkrétní struktura aplikace *xterm* je taková, že na nejvyšší úrovni vedle okna s emulací terminálu VT100 mohou být ještě okno grafického výstupu Tektronix a menu. Posuvné lišty se vztahují k oknu VT100. Uživatel, když atribut zapisuje, musí přesně znát hierarchii přípravků v konkrétní aplikaci. Tyto informace by měl nalézt v manuálové stránce aplikace (např. `man xterm`). X Window nám však nabízí zjednodušený zápis pomocí hvězdičkové konvence, např.:

```
xterm*scrollBar: True
```

V tomto konkrétním případě jsme se částečně odstínili od znalosti hierarchie přípravků v aplikaci *xterm*. Hvězdička zde představuje libovolný (tedy i nulový) počet komponent (objekt a subobjekty) jména atributu. Je významný rozdíl mezi interpretací hvězdičky na příkazovém řádku shellu a zde. Na příkazovém řádku shellu se hvězdička expanduje na libovolný počet znaků, zde na libovolný počet celých jmen komponent. Proto *nelze* hvězdičku použít v případě, že bychom pro aplikace *xcalc*, *xclock* a *xclipboard* chtěli nastavit obrácené zobrazování následovně:

```
xc*reverseVideo: True
```

Tady by se nastavení atributu vztahovalo k neexistující (?) aplikaci *xc*. Hvězdičku lze v zápisu atributu použít i více než jednu, lze je použít i společně s tečkami. Všeobecně se doporučuje na místě tečky používat hvězdičku. Můžete se tak vyvarovat problémů s drobnými nekompatibilitami při povýšení verze aplikace.

Instance a třída

Každá komponenta, z níž se atribut skládá, patří určité *třídě* (Class). Třída je tu k tomu, aby obsahovala více komponent. Např. již zmíněný *xterm* obsahuje třídu `Foreground` obsahující barvu textu (`foreground`), barvu ukazovátka a barvu textového kurzoru. Všechny tyto komponenty jsou definovány jako *instance* třídy `Foreground`. Pokud všem těmto atributům chceme nastavit tmavě modrou barvu, potom zadáme:

```
xterm*foreground: darkblue
xterm*cursorColor: darkblue
xterm*pointerColor: darkblue
```

nebo totéž provedeme nastavením třídy:

```
xterm*Foreground: darkblue
```

Třídu od konkrétní instance odlišuje úvodní velké písmeno. Podle konvence začínají instance malým písmenem a třídy velkým písmenem. Pokud je název instance nebo třídy tvořen více než jedním slovem, začíná každé další slovo velkým písmenem.

Aplikace *xterm* je jednoduchá na to, aby se na ní dala demonstrovat skutečná "moc" atributů. Ukažme si ještě jeden příklad. Mějme hypotetickou aplikaci *xclient*, která má pole tlačítek, v tomto poli mají být všechna tlačítka modrá s výjimkou jednoho tlačítka, které má být červené. Atributy potom zapíšeme např. následovně:

```
xclient*buttonbox*Buttons*foreground:
```

```
blue
xclient*buttonbox*delete*foreground:
red
```

Třída `Buttons` zahrnuje všechna tlačítka a `delete` je instance jednoho konkrétního tlačítka. Každý intuitivně cítí, že nastavením instance bude změněna barva jednoho tlačítka, i když vybarvení tohoto tlačítka bylo spolu s jinými již definováno. V krajním případě můžeme také napsat:

```
*Foreground: blue
```

Tím by měly být nastaveny všechny barvy popředí všech klientů na modrou barvu. Jména konkrétních tříd a instancí opět hledejte v manuálových stránkách příslušných aplikací. Často však zjistíte, že jméno třídy je identické se jménem instance (jediný rozdíl je ve velikosti počátečního písmena). To potom může znamenat, že třída je tvořena jedinou instancí. Může však také jít o jméno primární instance ve třídě s tím, že existují další podružné instance.

Nahrazovací znak ?

Počínaje verzí X Window R5 lze spolu s hvězdičkou používat další nahrazovací znak - otazník. Tento reprezentuje *právě jednu komponentu* (objekt, subobjekt) jména atributu. Ukažme si použití otazníku na příkladě:

```
xclient.?.?.Background: whitesmoke
```

Tento atribut nastaví barvu pozadí všech přípravků aplikace *xclient*, které v hierarchii stojí na úrovni vnuka (*aplikace.otec.syn.vnuk...*). Typickými představiteli vnuka jsou dialogové boxy, menu apod. Možný je i tento zápis:

```
xclient.?.?*Background: whitesmoke
```

V tomto případě se nastaví barva pozadí všech přípravků, které jsou na úrovni vnuka a na všech úrovních nižších. Zdůrazněme, že otazník představuje právě jednu komponentu a hvězdička žádnou, jednu nebo více komponent.

Priority při vyhodnocování zápisu atributu

Atributy se v systému definují na více úrovních. Ale i kdyby byly v jednom souboru, může i tam docházet ke konfliktům. Proto existují pravidla, která tyto konflikty řeší. Obecně platí, že přesnější zápis má přednost:

- nezáleží na pořadí definice atributů,
- definice instancí má přednost před definicí tříd,
- zápis s vyjmenováním komponent má přednost před zápisem s použitím '?' a '*',
- zápis s použitím '?' má přednost před zápisem s použitím '*'.

Připomeňme předchozí příklad, ve kterém červená barva pro instanci tlačítka `delete` měla přednost před modrým vybarvením třídy `Buttons` bez ohledu na pořadí obou definic. Uveďme ještě jeden příklad pro vyhodnocení atributů; řádky jsou seřazeny podle priority od nejnižší po nejvyšší:

```
*scrollBar: True
?*scrollBar: True
XTerm*scrollBar: True
xterm*scrollBar: False
```

Zadávání atributu při spuštění aplikace

Standardní X klient má volbu `-xrm`, pomocí které lze k existující databázi atributů přidat atribut další.

Např.:

```
xterm -xrm 'xterm*pointerShape: pirate'&
```

Atribut se přidá k existujícímu obsahu databáze atributů a případné kolize se vyřeší podle uvedených prioritních pravidel. Proto se neuplatní takto zadaný atribut:
`xterm -xrm '*pointerShape: gumby' &`
Poznamenejme, že atribut musí být uzavřen do dvojice apostrofů.

Význam volby `-name`

Opět každý standardní X klient rozpoznává volbu `-name`, pomocí ní z lze změnit jméno instance konkrétní spouštěné aplikace, například:

```
xterm -name bigxterm &
```

V tomto případě bude takto spuštěná aplikace akceptovat atributy určené třídě `XTerm` a instanci `bigxterm` (nikoli `xterm`). Proto můžeme mít atributy definované tímto způsobem:

```
XTerm*Font: 8x13
bigxterm*Font: 14x26
```

Spustíme-li *xterm* bez uvedení volby `-name` nebo s volbou `-name`, ale s jiným jménem než `bigxterm`, použije se font 8x13. Pokud *xterm* spustíme podle předchozího příkladu, použije se font 14x26.

Zadávání atributů X serveru

Atributy uživatel zapíše do souboru. Vždy jeden atribut na jeden řádek. Pro snazší orientaci v souboru smí použít *komentáře* a *pokračovací řádky*. Komentář se uvozuje znakem vykřičník (!). Definice, která bude mít pokračování na dalším řádku, se ukončí obráceným lomítkem (\).

Soubor s uloženými definicemi atributů může mít libovolné jméno. Atributy X serveru předává klient *xrdb*. Zpravidale se tento klient spouští při inicializaci X systému pro daného uživatele. Implicitně se definice čtou ze souboru `.Xresources` nebo `.Xdefaults`; ten se nejprve hledá v systémovém adresáři X Window (např. někde pod `/usr/X11`) a potom v domovském adresáři uživatele.

Klientu *xrdb* se při spuštění vedle jména souboru s atributy zadávají také následující volby: Volbou `-load` zajistíme, že před zaváděním atributů se předchozí nastavení zruší. Tím se ubezpečíme, že námi zavedené atributy budou jediné. Tato varianta je implicitní.

Volbou `-merge` docílíme přidání našich atributů k atributům v X serveru již existujícím. Při kolizích se postupuje podle dříve popsaných pravidel pro vyhodnocování priorit. Například:

```
$ xrdb -load .Xresources
```

Momentální obsah databáze atributů klient *xrdb* vypíše po uvedení volby `-query`. Máme také možnost uložit momentální obsah databáze do souboru se zachováním komentářů v souboru např. příkazem:

```
$ xrdb -edit .Xresources
```

Klientu *xrdb* můžeme zadat ještě řadu dalších voleb. Jejich popis najdete např. ve výpisu `man xrdb`.

Výpis atributů týkajících se klienta

Pro snadnější přístup k atributům patřícím určité třídě nebo instanci máme X klienta *appres* (*app*lication *res*ource). Jako parametr zadáváme třídu a příp. i instanci. Zjistíme tak, jaké atributy by aplikace určená třídou a instancí obdržela. Ukažme si použití *appres* na příkladech:

```
appres XTerm
appres XTerm bigxterm
appres XTerm -name bigxterm
```

V prvním případě se vypíší všechny atributy určené třídě `XTerm` a ve druhém a třetím případě pouze ty, které se vztahují současně ke třídě `XTerm` a k instanci `bigxterm`.

Další Předchozí Obsah