Ressourcenverwaltung unter X-Window

Wozu Ressourcen?

Jede X-Anwendung zeichnet sich durch eine mehr oder weniger umfangreiche Anzahl von Eigenschaften aus, die ihr Verhalten oder ihr Aussehen charakterisieren. Allein die xterm-Terminalemulation besitzt über hundert steuerbare Parameter (Existenz eines Scroll-Balkens, verwendeter Font, Vorder- und Hintergrundfarben, Geometrie ...). Die Frage ist, wie man diese Eigenschaften den persönlichen Wünschen u. Erfordernissen anpassen kann. Sie in den Kommandozeilen als Aufrufoptionen einzusetzen, stellt eine schwer zu handhabende Aufgabe dar (xterm hat über sechzig Optionen). Zum anderen unterstützt der größte Teil der X-Anwendungen meist nur die gängigsten Aufrufparameter, wie Größe u. Lage (-geometry) oder den verwendeten X-Server (-display) - obwohl viel mehr Einstellungen möglich sind (z.B. hat der Motif Windowmanager V1.1 fünf Kommandozeilen-Optionen u. über hundert beeinflußbare Eigenschaften).
Ein Mechanismus zur Bewältigung dieses Problems wird in X11 in der Form der X-Ressources bereitgestellt - das sind Eigenschaftsbeschreibungen von Anwendungen in der Form:

Anwendungseigenschaft: *Wert.*

Diese werden, nach unterschiedlichen Gesichtspunkten gruppiert, in system- oder benutzerdefinierten Dateien untergebracht. Die Anwendungen benutzen diese Dateien in ihrer Startphase zur Initialisierung auf dem jeweiligen X-Server. Die Änderung eines Ressourcenwertes für eine bereits laufende Anwendung wird von ihr in der Regel nicht bemerkt und wird erst nach einem 'Restart' wirksam. Dem Anwender ist es mit der Kenntnis der Ressourcen-Verwaltung möglich, sich auch in äußerst komplexen und wechselnden Umgebungen einen stabilen und maßgeschneiderten Arbeitsrahmen zu schaffen.

Wie sind Ressourcen definiert?

Syntax
Anwendungen in X-Window sind meist sehr komplex aufgebaut (Textbereiche, Scroll-Balken, Schalter, Menüs ...). Um diese einzelnen Elemente beschreiben zu können, wird die *Anwendungseigenschaft* in mehrere Teile strukturiert. Diese Teile werden durch den Punkt '.' als Trennzeichen miteinander verbunden.

Die Benennung einiger Bestandteile kann auch zugunsten eines '*' als Wildcard entfallen - der '*' substituiert dann alle möglichen Elemente (auch Kombinationen), die an seiner Stelle passen könnten.

Die Beschreibung einer *Anwendungseigenschaft* zerfällt logisch in drei Bereiche:
- der Name des Programms (xterm, xman ...); er kann durch einen '*' ersetzt sein
- eine oder mehrere Einschränkungen auf einen bestimmten Teil der Anwendung (menu, title, scrollbar ...); auch dieser ist durch '*' ersetzbar
- das zu charakterisierende Attribut (font, geometry, foreground ...); dieser Bestandteil muß vorhanden sein

Somit erfolgt eine Ressourcen-Definition in der Form:

[*Programmname*][*Einschränkung*...]*Attribut*: *Wert*

Beispiele für gültige Festlegungen sind:

xterm.vt100.scrollbar.foreground: red
- legt die Vordergrundfarbe des Scroll-Balkens eines xterm-Windows mit rot fest; Beispiel für einen vollständigen Ressourcennamen

xterm.vt100.foreground: brown
- legt die Textfarbe des xterm's fest

xterm*foreground: blue
- legt die Vordergrundfarbe für Text und Scroll-Balken auf blau fest, da durch den '*' beide erfaßt werden

*font: 8x13
- legt den Font für alle X-Anwendungen ohne jede Restriktion auf den Typ 8x13 fest, sofern nicht konkretere Festlegungen etwas anderes bestimmen (siehe unten)

Klassen und Instanzen
X-Anwendungen sind ihrer Herkunft nach Produkte objektorientierter Programmierung. Vereinfacht kann man sich vorstellen, daß in dieser Programmiertechnik einzelne Programmodule erstellt werden, die nach außen eine abstrakte Datenstruktur und die auf sie erklärten Funktionen definieren. Zum Beispiel wird im X-Toolkit ein Typ 'Font' und die dazugehörigen Funktionen 'XLoadFont' (lädt einen Font), 'XSetFont' (setzt Fontattribute) und 'XUnloadFont' (entfernt Font) definiert, der im Programmodul 'VT100' (seinerseits ein abstrakter Datentyp) als Variable 'font' vom Typ 'Font' benutzt wird. Die Funktionen spielen für den Nutzer keine weitere Rolle - die Datenstrukturen spiegeln sich jedoch in den Ressourcen wieder. Daher haben alle Elemente einer *Anwendungseigenschaft* im Sinne der X-Ressourcen eine Doppelbedeutung. Zum einen stellen sie einen allgemeinen Typ (die Klasse)

dar, der von der nächst höheren Ebene genutzt wird. Zum anderen gibt es von diesen Typen Variablenrealisierungen - die Instanzen. Klassen u. Instanzen können in den X-Ressourcen (auch gemischt) vorkommen, wobei Klassen im allgemeineren und Instanzen im konkreteren Sinne gebraucht werden. Klassennamen beginnen mit großen, Instanzennamen mit kleinen Buchstaben. Zum Beispiel ist 'xterm' die Instanz der Klasse 'XTerm', 'mwm' die Instanz von 'Mwm', 'scrollbar' von 'Scrollbar' u. 'font' von 'Font'.

Wie erhält man nun die Namen von Klassen und Instanzen eines X-Programmes?
Zuallererst sollte man in den Manuals nachblättern. Sollte dies nicht erfolgreich sein, dann suche man sein Glück im Verzeichnis /usr/lib/X11/app-defaults. Dort findet man Dateien mit dem Klassennamen von X-Programmen wie 'Xman', 'Mwm', .. Diese Dateien enthalten mehr oder weniger üppig die systemweiten Ressourcen-Voreinstellungen dieser Programme.
Die Instanzennamen der Programme sind ihre Namen selbst. X-Toolkit-Anwendungen können ihren Instanzennamen mit dem Parameter '-name' ändern. Zum Beispiel bewirken die Aufrufe

```
xterm&
xterm -name hugo&
```

mit den Ressourcen-Voreinstellungen

```
XTerm*background:   yellow
xterm*title:        Ich bin ein xterm
xterm*cursorColor:  blue
hugo*title:         Ich bin Hugo
hugo*cusorColor:    red
```

daß das Window mit 'Ich bin ein xterm' in der Überschrift einen blauen Textkursor besitzt, während das 'Ich bin Hugo'-Window über einen roten verfügt. Beide haben als Mitglieder der Klasse XTerm einen gelben Hintergrund.
Ein Beispiel, das zeigt, wieviel Instanzen eine Klasse umfassen kann, demonstriert die Klasse 'Foreground'. Für die Zeile

XTerm*Foreground: blue

müßten folgende Instanzen verwendet werden:

```
XTerm*foreground: blue      - Textfarbe
XTerm*cursorColor: blue     - Textkursorfarbe
XTerm*pointerColor: blue    - Mauskursorfarbe
XTerm*Scrollbar.foreground: blue
                            - Scrollbalkenfarbe
```

Das heißt, die Attribute 'foreground', 'cursorColor', 'pointerColor' und 'foreground' (in der Klasse Scrollbar) sind Instanzen der Klasse 'Foreground'.

Vorrangregeln
Wenn zwei Spezifikationen die gleiche *Anwendungs-eigenschaft* beschreiben, dann gilt ganz allgemein, daß diejenige zur Wirkung kommt, die die Eigenschaft genauer (spezifischer) beschreibt. Folgende Regeln kommen dabei zum Tragen:
- Der Punkt als Trennzeichen ist spezifischer als der Stern.
  XTerm.iconName ist spezifischer als XTerm*iconName
- Instanzennamen sind spezifischer als Klassennamen.
  *foreground ist spezifischer als *Foreground
- Eine vorhandene Komponente (Einschränkung) ist spezifischer als ihr Weglassen.
  XTerm*scrollBar.foreground ist spezifischer als XTerm*foreground
- Komponenten weiter links in der Spezifikation sind spezifischer als weiter rechts stehende.
  XTerm*vt100*foregroundist spezifischer als XTerm*scrollbar*foreground

Allgemein kann man sagen: Ist eine Spezifikation nur eine Teilmenge einer vorhandenen, dann ist die erste spezifischer als die zweite.

Wie können Ressourcen verwaltet werden?

Die Verwaltung der Ressourcen stellt für den weniger erfahrenen X-Nutzer ein Problem dar. Einträge können sich in zehn verschiedenen Dateien befinden, fünf Environment-Variablen wollen beachtet sein und einer so geheimnisvollen Einrichtung wie dem RESOURCE_MANAGER Property kommt eine zentrale Bedeutung zu. Ohne Kenntnis der Zusammenhänge kann man sich leicht in diesem Labyrinth verirren - überraschende Effekte und zeitraubendes Suchen sind das Resultat. Ziel des Kapitels ist es, Systematik in dieses Chaos zu bringen und beim Leser für die grundlegenden Konzepte Verständnis zu wecken.

Der Ressourcen-Suchpfad
Von zwei Ausnahmen abgesehen befinden sich in diesem Suchpfad nur Dateien. Sie enthalten entweder die Ressourcen einer speziellen Anwendung oder sind Sammelbecken der Anwendungseigenschaften verschiedener Programme. Als Kommentarzeichen wird in ihnen das Ausrufezeichen '!' verwendet. Eine Anwendung bewegt sich in fünf Etappen durch den Ressourcen-Irrgarten (siehe Abbildung). Jeden Block durchsucht sie von oben beginnend nach dem Vorhandensein der ersten gültigen Ressourcen-Quelle. Ist die Anwendung fündig geworden, wertet sie diese Quelle aus u. geht zum nächsten Block (weitere vor-handene Quellen in diesem Block werden ignoriert).

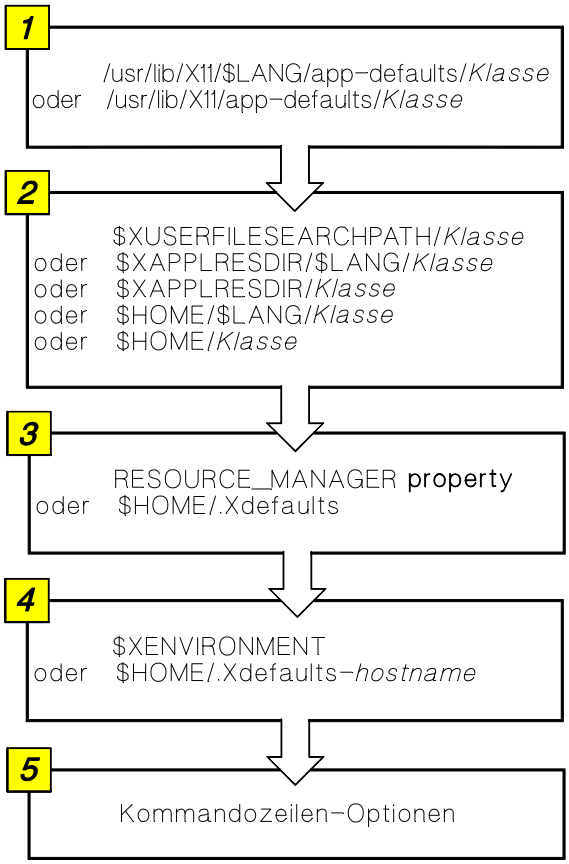

Abb.: Ressourcen-Quellen für Anwendungen

Werden an unterschiedlichen Orten gleiche Ressourcen-Spezifikationen mit abweichenden Werten belegt, dann ist die weiter unten stehende gültig (besitzt eine höhere Priorität).
Vorsicht - die zuletzt getroffene Aussage gilt nur für identisch formulierte Ressourcen. Treten nur geringfügige Unterschiede auf, dann werden Vorrangregeln wirksam, die die Priorität erheblich beeinflussen können.
Steht in der Datei $HOME/XTerm (Block2):

xterm.vt100.foreground: blue

und in $HOME/.Xdefaults (Block3):

xterm.vt100*foregound: red

dann wird die Textfarbe eines xterm-Windows blau, da Punkt vor Stern kommt, obwohl der Rot-Eintrag weiter unten steht. Dafür wird der Scrollbalken rot, da im Block2 darüber nichts ausgesagt wird u. er in Block3 mit dem Stern erfaßt wird.
Die verwendeten Environment-Variablen werden in der Regel nicht vom System bereitgestellt u. unterliegen vollständig der Nutzerkontrolle. Noch ein Wort zur Environment-Variablen $LANG. Sie dient der Anpassung von Anwendungen an nationalsprachliche Gegebenheiten (z.B. deutschsprachige Tastatur, Wahl eines iso8859-1 Zeichensatzes). Dem Nutzer eröffnet sie die Möglichkeit, seine Sitzungen sowohl auf eine 'deutsche' Konsole als auch auf ein 'englisches' X-Terminal abzustimmen. Statt der Variablen $LANG kann auch der Ressourcen-Parameter *xnlLanguage verwendet werden.

(1) Anwendungsspezifische Ressourcen
In diesem Block werden Herstellervorgaben für das Programm formuliert. Die Dateien tragen den Klassennamen der Anwendung (XTerm, Mwm, ...) u. bilden bei schlecht dokumentierter Software oft den einzigen Fundus für Ressourcenspezifikationen. Der Inhalt dieser Dateien sollte möglichst nicht verändert werden.

(2) Benutzerspezifische Ressourcen
Hier erhält der Nutzer die Möglichkeit, die Anwendungseigenschaften den eigenen Bedürfnissen anzupassen. Er kann das gleich in seinem HOME-Verzeichnis tun oder aber mittels Environment-Variablen ein anderes Quellverzeichnis festlegen. In der Literatur wird erwähnt, daß der Pfadeintrag in $XAPPLRESDIR mit einem abschließenden Schrägstrich enden muß (z.B. /home/muster/source/), aber zumindest auf SUN-, IBM u. DEC-Rechnern ließ sich das nicht bestätigen - bei ihnen funktionierte der Eintrag sowohl mit als auch ohne endenden Strich.

(3) Serverspezifische Ressourcen
An dieser Stelle können alle die Eigenschaften der Anwendungen definiert werden, die auf einen speziellen X-Server ausgerichtet sind. Zum Beispiel haben Größe und Auflösung des Bildschirmes Einfluß auf die verwendete Fontgröße. Das übliche Verfahren besteht darin, auf dem X-Server via RESOURCE_MANAGER Property eine Ressourcen-Datenbank zu errichten (siehe unten). Eine eher traditionelle Methode ist die Nutzung der Datei $HOME/.Xdefaults.

(4) Rechnerspezifische Ressourcen
In diesem Block wird zuerst der Inhalt der Environment-Variablen $XENVIRONMENT ausgewertet. Diese weist mit vollem Pfadnamen auf eine Datei mit Ressourcen-Festlegungen (im Gegensatz zu $XAPPLRESDIR, in der nur der Pfad auf ein Directory steht). Ist diese Variable nicht definiert, dann sucht die Anwendung nach einer Datei .Xdefaults-*hostname*, wobei *hostname* der Name des Rechners ist, auf dem das Programm läuft. Das spielt in Nutzerumgebungen mit einem Netzwerk-Dateisystem (NFS) eine Rolle, wo das Heimatverzeichnis auf einem anderen als dem Rechner liegt, auf dem das Programm startet oder der X-Server läuft.

(5) Kommandozeilen-Optionen

Die höchste Priorität genießen die Vorgaben aus der Kommandozeile. Wenn 'xterm -sb' aufgerufen wurde, dann erscheint der Scrollbalken eines xterm's, egal ob in irgend einer anderen Quelle abweichendes festgelegt wurde. Viele Anwendungen unterstützen einen Aufrufparameter -xrm, mit dem Ressourcen-Einstellungen über die Kommandozeile eingegeben werden können. Doch sollte hier beachtet werden, daß Vorrangregeln die eigentliche Absicht zunichte machen können. Enthält zum Beispiel die Datei .Xdefaults folgende Zeile

xterm*scrollBar: false

und man versucht einen Scroll-Balken mit

xterm -xrm "XTerm*ScrollBar: true" &

zu erzeugen, dann verhindert das die Vorrangregel, da Instanzen spezifischer sind als Klassen.

X-Ressourcen-Datenbank

Da praktisch alle Ressourcenwerte auf den X-Server wirken (Farben, Fonts, Maus- und Tastaturverhalten) und Anwendungen von allen Rechnern im Netzwerk auf ihm agieren können, ist es nur logisch, die Ressourcendefinitionen zentral auf dem X-Server zu führen. Über die "Properties" wird dem Nutzer eine Möglichkeit in die Hand gegeben, auf dem X-Server eine Ressourcen-Datenbank anzulegen. Was nun eigentlich Properties sind und wie sie verwaltet werden können, wird im folgenden ausgeführt.

Properties

Mit den Properties stellt X-Window einen Mechanismus für die Kommunikation zwischen verschiedenen X-Anwendungen zur Verfügung. Properties sind Datenpakete, die die Anwendungen an beliebige Windows hängen können, aber auch jederzeit lesen können - sie sind lokal, wenn sie an Anwendungen hängen bzw. global, wenn sie im Root-Window ruhen. Properties sind die Environment-Variablen des X-Window-Systems, mit dem Unterschied, daß sie nur sehr begrenzt der Nutzerkontrolle unterliegen. Mit dem Kommando 'xprop' kann man sich die Properties eines jeden Windows anzeigen lassen. Jede Anwendung überprüft beim Start, ob im Root-Window ein Property namens 'RESOURCE_MANAGER' existiert und wenn ja, dann fischt sie die für sich passenden Einträge heraus. Die Manipulation dieses Propertys mit Hilfe des Programms 'xrdb' wird im folgenden Abschnitt diskutiert.

Ressourcenverwaltung mit xrdb

Mit dem Programm xrdb (**X R**esource **D**ata**B**ase utility) kann man

- das RESOURCE_MANAGER Property generieren
  xrdb *Datei*
  (Achtung - eine bestehende Datenbasis wird damit vollständig überschrieben !)
- sich deren Inhalt anzeigen lassen
  xrdb -query
- den Inhalt in eine *Datei* schreiben
  xrdb -edit *Datei*
- neue Einträge hinzufügen
  xrdb -merge *Datei*
  (die Ressourcen-Festlegungen der *Datei* werden der bestehenden Datenbasis hinzugefügt)
- das Property komplett löschen
  xrdb -remove

Wird *Datei* weggelassen, dann liest xrdb von der Standardeingabe. Wenn man bei der Inhaltsanzeige den Parameter '-query' vergißt (also nur 'xrdb' aufruft), dann hat das den unangenehmen Effekt, daß xrdb die Datenbasis löscht und über den Eingabekanal auf neue Ressourcen wartet. Es gibt keinen von xrdb unterstützten Weg, in einem vorhandenem RESOURCE_MANAGER Property Veränderungen durchzuführen. Man kann sich aber mit folgender Sequenz behelfen:

```
xrdb  -edit  /tmp/xrdb.res
vi  /tmp/xrdb.res
xrdb  /tmp/xrdb.res
rm  /tmp/xrdb.res
```

Eine hervorstechende Eigenschaft von xrdb besteht darin, daß er die Eingabe-*Datei* standardmäßig durch den C-Präprozessor filtern läßt. Dafür stellt er eine Reihe von Präprozessorsymbolen zur Verfügung. Man kann sich diese Symbole und deren momentane Inhalte mit

xrdb -symbols

anzeigen lassen (man sieht dann die Aufrufparameter, die xrdb dem C-Präprozessor übergibt). Sie sollten dafür benutzt werden, die Ressourcen-Einstellungen der momentanen X-Server-Umgebung anzupassen. Wechselt man öfter von Farb- zu Schwarz-Weiß-X-Terminals, dann kann folgendes in der an xrdb übergebenen *Datei* stehen:

```
#ifdef COLOR
XTerm*Foreground: blue
XTerm*Background: yellow
#endif
! ansonsten laß es bei schwarz-weiß
```

Die wichtigsten Symbole sind:

HOST — gibt den Rechnernamen aus der Environment-Variablen $DISPLAY an

CLIENTHOST — bezeichnet den Rechner, von dem xrdb aufgerufen wurde

WIDTH, HEIGTH — Breite und Höhe des Bildschirms in Pixeln

X_RESOLUTION, Y_RESOLUTION

	x- und y- Auflösung des Bildschirms in Pixel pro Meter
PLANES	Anzahl der Ebenen für Graustufen- oder Farbdarstellung
COLOR	ist nur bei Farbbildschirmen definiert

Die Präprozessorsymbole können an jeder Stelle der *Datei* auftreten, nicht nur in den #-Zeilen. Will man eine Anwendung über den ganzen Bildschirm ziehen

xload*Width: WIDTH

ergibt sich bei einem 1280x1024-Bildschirm nach der Präprozessorauswertung

xload*Width: 1280.

Abrundendes Beispiel

Ausgangsposition
Folgende Rechner befinden sich im lokalen Umfeld eines Nutzers:

- ein Compute- und File-Server mit hoher Verarbeitungs- und Plattenkapazität, aber ohne eigene X-Geräte (mit Namen 'maximus')
- eine Workstation mit 17"-Farbbildschirm und der Auflösung 1280x1024 bei 256 Farben ('smiley')
- ein PC mit 14"-Farbbildschirm und der Auflösung 800x600 bei 16 Farben ('mini')

Über ein Netzwerk sind alle miteinander verbunden.

Folgende Besonderheiten sind für die beiden X-Server-Maschinen kennzeichnend:
smiley
Die Workstation verfügt über nicht ausreichend Plattenplatz, um Nutzerdateien zu halten - so importiert sie via NFS die Nutzerverzeichnisse von **maximus** (d.h. man loggt sich zwar auf **smiley** ein, kann seine Programme nutzen, landet aber mit seinem Heimatverzeichnis auf **maximus**). Auf **smiley** läuft ein lokaler xdm (ein Programm, das für

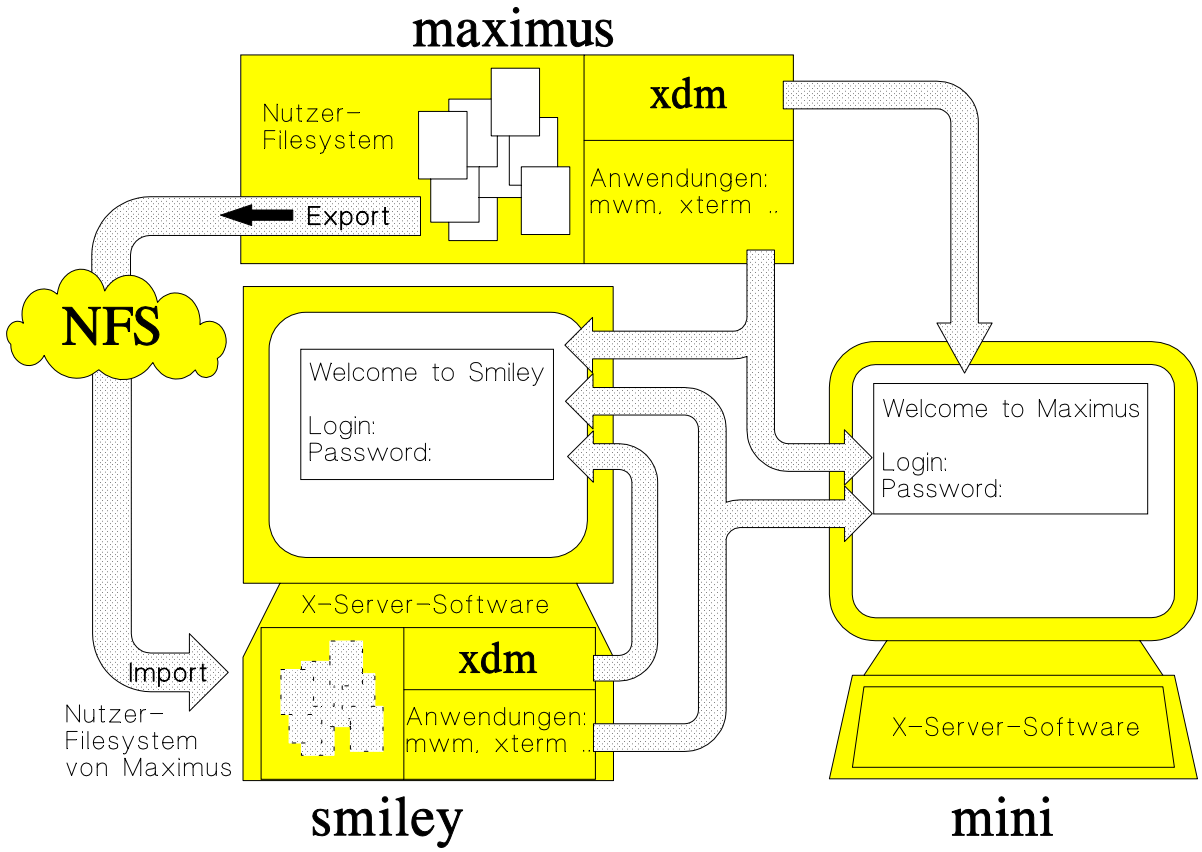

Abb.: Beispielkonfiguration für eine X-Window-Umgebung in einem Netzwerk
Die Pfeile charakterisieren, welche Programme von welchem Rechner auf jeweils welche Maschine wirken könne

den Sitzungsverlauf auf einem X-Server verantwortlich zeichnet), der den eigenen X-Server verwaltet.

mini

Auf **maximus** versorgt der xdm den entfernten X-Server **mini** mit einem X-Login, wenn auf ihm eine X-Server-Emulation (XVision, XView, ..) in Betrieb ist. Vom PC aus loggt man sich also direkt auf **maximus** ein und hat die Programmbibliothek von **maximus** zur Verfügung.

Aus dieser Konstellation (2 X-Server - **smiley**, **mini,** 2 Login-Rechner - **smiley**, **maximus** und ein und dasselbe Heimatverzeichnis) ergibt sich das Problem, die Initialisierungsdateien im Heimatverzeichnis von **maximus** so zu formulieren, daß ein vernünftiges Arbeiten an beiden X-Bildschirmen möglich wird.

Realisation

Vor der Betrachtung der Ressourcen sei die StartUp-Datei $HOME/.xsession (wird von xdm zur Initialisierung der X-Sitzung genutzt) vorangestellt:

```
#!/bin/sh
# Ermitteln Hostname; Domainname entfernen
CPU=`hostname | sed -e 's/\..*//'`

# Für rsh-Rufe $DISPLAY auf smiley korrigieren
case $DISPLAY in
  unix:0.0|unix:0|:0.0|:0) DISPLAY="$CPU:0";;
esac; export DISPLAY

# Aktivierung der Ressourcen für X-Server
xrdb $HOME/.Xdefaults

xhost +        # Zugriff auf X-Server gestatten
mwm&           # Windowmanager starten

# von jeweils anderem Rechner xterm holen
case $CPU in
 smiley) rsh maximus xterm -display $DISPLAY \
                -geometry -0-0 < /dev/null &;;
 maximus) rsh smiley xterm -display $DISPLAY \
                -geometry -0-0 < /dev/null &;;
esac

# abschließend Login-Window setzen
exec xterm -name login -geometry +0+0
```

Diese Datei .xsession bewirkt unabhängig vom verwendeten Rechner, an dem man sitzt, daß auf dem X-Bildschirm links oben das Login-xterm-Window des Heimatsystems[1] und rechts unten ein xterm des jeweilig anderen Rechners erscheint. Außerdem wird der lokale Windowmanager gestartet.

Noch ein Wort zur Korrektur von $DISPLAY auf **smiley**. Die X-Initialisierung auf Workstations mit eigenem X-Bildschirm erzeugt in der Regel einen $DISPLAY-Wert von 'unix:0' oder ähnlich. Dies hat für die Workstation-Programme den Vorteil, daß sie nach Erkennen eines solchen Wertes wissen, daß der X-Server sich auf der lokalen Maschine befindet und unter Vermeidung der Netzwerkschichten erreichbar ist. Im Netzwerk ist $DISPLAY dann allerdings wertlos, weil aus ihr nicht mehr der Ort des X-Servers ermittelbar ist. Er wird für **smiley** berichtigt, damit er acht Zeilen tiefer in einem rsh-Aufruf benutzt werden kann. Der xdm von **maximus** dagegen setzt den $DISPLAY-Wert für den entfernten X-Server **mini** richtig, und dieser wird dann durch die case-Anweisung nicht berührt.

Damit die Remote-Shell-Aufrufe an dem jeweilig anderen Rechner auch gelingen, sollte die Datei $HOME/.rhosts mit folgenden Einträgen existieren:

```
smiley
maximus
```

Sie ermöglicht die Nutzung von rsh; sollten aber die Nutzernamen auf den beiden UNIX-Rechner unterschiedlich sein, so müssen diese sowohl beim Aufruf als auch in dieser Datei berücksichtigt werden (siehe Manual).

Nun muß das Problem gelöst werden, wie man sich die Umgebung anpaßt, um auf beiden X-Servern vernünftig arbeiten zu können.

Zuerst fasse man die Eigenschaften zusammen, die pauschal für beide gelten sollen (unabhängig davon, woher die Anwendung kommt und auf welchem X-Server sie dargestellt werden soll). Das könnte für das Programm xterm in der Datei $HOME/XTerm etwa so aussehen:

```
*ScrollBar:          true
*SaveLines:          200
*LoginShell:         true
*ReverseWrap:        true
```

Hier ist es nicht nötig, die Klasse am Anfang aufzuführen, da diese Datei ohnehin nur von xterm's konsultiert wird. In dieser Datei sollten keine Festlegungen enthalten sein, die in irgendeiner Weise Eigenschaften des X-Servers berühren (keine Font-, Farben- oder Geometriefestlegungen). Einige Einträge für $HOME/Mwm (Motif-Windowmanager) könnten so aussehen:

```
*KeyboardFocusPolicy: pointer
*InteractivePlacement:  true
*login*IconImage:       $HOME/bitmaps/terminal
*login*ClientFunctions: -close
*UseIconBox:            true
```

Hier wird für die Anwendung mit dem Namen 'login' (mit der Option '-name login' gestartet - siehe .xsession) ein spezielles Icon definiert. Außerdem kann das Login-xterm nicht mehr mit Windowmanager-Mitteln beendet werden.

Für den serverabhängigen Teil der Ressourcen-Beschreibung wird im Initialisierungsprogramm .xsession die Datei $HOME/.Xdefaults benutzt, um

[1] Das System, an dem man sich einloggt - an der Konsole von **smiley** ist es die Workstation **smiley,** und am PC **mini** ist es die Maschine **maximus**

eine Ressourcen-Datenbank aufzubauen. Sie könnte folgendes Aussehen haben:

```
#if HOME==smiley
#include ".Xresources-smiley"
#endif
#if HOME==mini
#include ".Xresources-mini"
#endif
```

In dieser Form werden für jeden der beiden X-Server eigene Beschreibungsdateien angelegt, in denen dann die Spezifika ausformuliert werden können. Die Datei $HOME/.Xresources-mini könnte dann etwa nachstehenden Anblick bieten:

```
XTerm*Font:              8x13
XTerm*BoldFont:          8x13bold
Mwm*FontList:            8x13
Mwm*Background:          gray
Mwm*ActiveBackground:    cyan
```

Diese Methode bietet den Vorteil, daß man für jeden Bildschirm, unabhängig vom anderen, Feineinstellungen für die eigene Umgebung vornehmen kann - nachteilig ist, daß man sich bei Veränderungen der X-Server (Austausch oder neu hinzukommender) wieder um seine Ressourcen kümmern muß.
Dieser Nachteil kann vermieden werden, wenn man in der Datei $HOME/.Xdefaults eine allgemeinere Darstellungsweise findet, die weitestgehend unabhängig vom verwendeten X-Server ist:

```
#if X_RESOLUTION>3500
XTerm*Font:     *-courier-medium-r-*-140-100-*
XTerm*BoldFont:  *-courier-bold-r-*-140-100-*
Mwm*FontList: *-courier-medium-r-*-140-100-*
#else
XTerm*Font:      *-courier-medium-r-*-140-75-*
XTerm*BoldFont:    *-courier-bold-r-*-140-75-*
Mwm*FontList:   *-courier-medium-r-*-140-75-*
#endif
#ifdef COLOR
Mwm*Background:            gray
Mwm*ActiveBackground:    cyan
#else
Mwm*Background:            25_foreground
Mwm*ActiveBackground:    75_foreground
Mwm*ActiveForeground:    white
#endif
```

In diesem Beispiel werden 100dpi- oder 75dpi-Fonts je nach Auflösung des X-Servers verwendet (dabei entsprechen 3000 Pix/m, etwa 75 dpi, dem PC **mini** und 4000 Pix/m, etwa 100 dpi, der Workstation **smiley**). Der Motif-Rahmen wird farbig oder schattiert je nach Farbmöglichkeiten. Auf diese Art und Weise ist man zwar relativ gefeit gegen Kontextwechsel, aber der letzte Feinschliff für einen speziellen X-Server läßt sich damit nur schwerlich realisieren.
Als nächstes kann man Ressourcen-Festlegungen für Anwendungen in Abhängigkeit von dem Rechner treffen, auf dem sie gestartet werden. Im Beispiel wären das **maximus** und **smiley,** und demzufolge hießen die zugehörigen Ressourcen-Dateien '.Xdefaults-maximus' und '.Xdefaults-smiley'. Eine typische Zeile daraus wäre:

```
! $HOME/.Xdefaults-smiley
XTerm*Title:     Smiley
```

und

```
! $HOME/.Xdefaults-maximus
XTerm*Title:       Maximus
```

Die Wirkung besteht darin, daß xterm's den Namen des jeweiligen Rechners in ihrem Motif-Rahmen führen.
Schließlich können Ressourcen-Werte auch über die Kommandozeile beeinflußt werden. Dies sollte jedoch nur im Ausnahmefall geschehen - sollte sich der gleiche Wert ständig wiederholen, dann sollte man ihn in einer der eben diskutierten Dateien definieren.

Schlußbemerkung

Der Ressourcen-Mechanismus ist ein enorm machtvolles und flexibles Instrument zur Anpassung der X-Umgebung an die Bedürfnisse des Nutzers. Das wichtigste an ihm ist aber, nicht den Überblick zu verlieren. Es gilt auch hier: Weniger ist mehr. Vor allem Wildcards und Klassen-Festlegungen können unerwartete Seiteneffekte produzieren. Als Anfänger auf diesem Gebiet sollte man zuerst mit Farb- oder Fontfestlegungen beginnen und sich mit zunehmender Sicherheit an komplexere Definitionen heranwagen. Wie gut oder schlecht man seine Ressourcen verwaltet, kann man letztlich auch daran messen, wieviel Zeitaufwand dafür verwandt wird.
In diesem Sinne - viel Spaß beim Experimentieren !

Frank Sittel