NAME

xtoolplaces - save current X desktop window information

SYNOPSIS

xtoolplaces [-a filename] [-c] [-display host:display.screen] [-f filename] [-m filename] [-r command] [-s {list | all}] [-v]

DESCRIPTION

Xtoolplaces collects current state information for all windows running on an X display at the time of execution. This information can then be saved and used to restart the X display. Applications started remotely are saved correctly. Local applications lines are of the form:

```
WM_COMMAND
```

while remote application lines are of the form:

```
rsh -n <machine> WM_COMMAND
```

WM_COMMAND is the command line to use to re-create the application as it currently exists and <machine> is the remote application's host. The '-r' option can be used to change the command used to start remote applications (i.e. rcmd or xon). If an application doesn't update WM_COMMAND, the geometry information will be gotten manually. Use the '-m' option to save those applications that don't set WM_COMMAND (i.e. xman).

Xtoolplaces will lock the pointer, change the cursor to a watch, and beep once to announce that it's starting. The pointer is locked so that the user can't change anything once xtoolplaces starts. When the program is finished it releases the pointer, changes the cursor back to the normal shape, and beeps twice.

The default filename to store the window information is ˜/.xtoolplaces. This can be overridden with the '-f' option (including writing to *stdout*).

This is the X equivalent of Sunview's toolplaces (but is not limited to Suns -- I hope). Xtoolplaces was written entirely in Xlib to make it as portable across X platforms as possible. This tool was born out of the need to replace the functionality of toolplaces in X. The Open Look window manager (olwm) could only save the window information into one place. In addition, it could only save applications that correctly saved their startup state in WM_COMMAND. Since I needed different initialization files depending on what machine I was on and I used applications that weren't "well-behaved", this became a little limited. Therefore, xtoolplaces was made.

OPTIONS

Descriptions of the '-a', '-c', '-m', and '-s' options are listed in greater detail in following sections. A description of resources is listed in the section entitled RESOURCES.

-a filename
: Read in a file ('filename') containing arguments to add on to the end of command lines. See ADDING ARGUMENTS below for more detail.

-c
: Put special 'if ... fi' statement around any console windows found. See CONSOLE CHECKING below for more detail.

-d host:display.screen
: Save window information from an alternate display.

-f filename
: Write window information to 'filename'. Default is '˜/.xtoolplaces'. Window information can be written to *stdout* by specifying '-' as the filename.

-m filename
: Read in a file ('filename') containing a list of X applications that don't set WM_COMMAND and need to be saved. Normally, X windows that don't have WM_COMMAND set aren't saved. See MISSING APPLICATIONS below for more detail.

-r command
: Command to use to start remote applications. Default is 'rsh -n'. The '-n' option of rsh is used to redirect input to /dev/null. See the rsh man page for more information on '-n'.

-s {list | all}
: Specify what screens to save information from. Either a list of screen numbers can be given or the keyword 'all' which specifies to do all the screens that a server knows about. See SCREEN LISTS below for more detail.

-v
: Print the current version number and patchlevel and quit. No processing is done.

ADDING ARGUMENTS (-a)

The '-a' option was added because of a possible xview problem. For some reason, some windows don't fill in the WM_COMMAND string completely. For example, xrolo is an xview program. When it saves it's state in WM_COMMAND, it never saves the filename it is using. This option allows a user to have a file containing the name of the program and any arguments to be added on to the end of the command line (gotten from WM_COMMAND). If multiple versions of the program are running, the extra arguments will go onto the end of the command line for each of them. If the same program is listed more than once in the file, only the first occurrence is used.

An addon file can consist of three different types of lines:

comment lines
blank lines
argument lines

Comment lines are lines where the first character is a pound sign (#).

Blank lines are lines where the first character is a newline.

Argument lines are of the form:

```
name: argument(s)
```

where 'name' is the name of the program whose command line you want to add onto and 'argument(s)' are the arguments to add. Everything from the first non-whitespace after to colon to the newline is considered arguments to add.

CONSOLE CHECKING (-c)

A console checking option was added out of a need to NOT start a console window if running from an X terminal without having a special initialization file for the terminal. This option will try to determine which windows (if any) are the console windows and put an 'if' statement around them to start only if the current tty equals '/dev/console'. This allows a user to have the same initialization file for starting X on the console of machine A and for an X terminal running off of machine A.

This version currently searchs for three types of console windows: *contool*, a console program to capture and display console output by Chuck Musciano; Sun's *shelltool* and *cmdtool* in console mode; and *xterm* in console mode, by Tom Weinstein and a whole bunch of other people. A console window is first checked by it's class. While there is no default class for console windows, maybe this program will create one. Xtoolplaces checks for a class of 'XConsole'. If the class doesn't match, it checks for the presence of the three above mentioned console windows. For *contool*, it checks it's window name. For *{shell,cmd}tool*, it checks the icon name. For *xterm*, it checks the options that xterm was started with.

MISSING APPLICATIONS (-m)

The ’-m’ option was added because some older X applications don’t set WM_COMMAND. Xtoolplaces uses the existance of WM_COMMAND to know what windows to save information from. The missing applications (to save despite not having WM_COMMAND set) file is a list of applications, one per line. Missing applications files have the same three line types as addons (see ADDING ARGUMENTS above) except that the argument lines consist of just the name (as gotten from WM_NAME) of the application to save. For example, we know that the programs *xman* and *xyz* don’t set WM_COMMAND but we wish to save their states anyway. The file specifed with ’-m’ would look like:

```
xman
xyz
```

SCREEN LISTS

A list of screens is a list of numbers seperated by spaces and quoted (by single or double quotes) to prevent the shell from parsing the list (i.e. to save screens 1 and 3 would be -s ’1 3’). The default screen to save is the one listed in either the environment variable *DISPLAY* or with the ’-d’ option. The screen specified by the ’-d’ option would override the one in *DISPLAY*.

When saving from multiple screens, each window saved has the ’-display’ option added to it if it doesn’t already have it. This is done in case a user starts a window from one screen (so as not to need -display) and runs xtoolplaces from another. To insure that the window starts up on the right screen, the ’-display’ option is added.

Three assumptions are made when adding the ’-display’ option:

```
1) if a colon (:) is found on the command line, it belongs
   to HOST:SERVER.SCREEN and not to some other option
2) if a window already has a ’-display’ option, it points
   to the correct screen and nothing is changed or added
3) every window that has ’-display’ added uses ’-display’
   (and not ’-d’ or something else)
```

RESOURCES

All of the options available on the command line can be put into a resource file (and in fact most are in the default resource file, /usr/lib/X11/app-defaults/Xtoolplaces), with the exception of the version (-v) flag. The order in which resources are read in (with each layer overwriting the previous layers) are as follows:

```
1) Hardcoded defaults
2) Values in /usr/lib/X11/app-defaults/Xtoolplaces
3) Values in $XUSERFILESEARCHPATH/Xtoolplaces or
   $XAPPLRESDIR/Xtoolplaces
4) Values set using xrdb, either through the
   XResourceMangerString macro or, if that’s empty, the
   .Xdefaults file in the user’s home directory
5) Values in the file specified by the XENVIRONMENT
   environment variable or, if that’s not set, the
   .Xdefaults-hostname file in the user’s home directory
   (where hostname is the name of the machine)
6) Command line arguments
```

Descriptions of each resource can be found in the OPTIONS section. The resource names are as follows:

xtoolplaces.addon
    -a option (i.e. xtoolplaces.addon: /adm/addon)

xtoolplaces.console
    -c option (i.e. xtoolplaces.console:)

xtoolplaces.display
    -d option (i.e. xtoolplaces.display: chroma:1)

xtoolplaces.saveto
    -f option (i.e. xtoolplaces.saveto: /home/abb/.startup)

xtoolplaces.missing
    -m option (i.e. xtoolplaces.missing: /adm/missing)

xtoolplaces.remote
    -r option (i.e. xtoolplaces.remote: rsh)

xtoolplaces.screens
    -s option (i.e. xtoolplaces.screens: 1 3 4)

FILES

/usr/lib/X11/app-defaults/Xtoolplaces	default resource file
$USERFILESEARCHPATH/Xtoolplaces	resource file
$XAPPLRESDIR/Xtoolplaces	resource file
˜/.Xdefaults	user's resource file
$XENVIRONMENT	resource file
˜/.Xdefaults-hostname	resource file for a specific machine
˜/.xtoolplaces	default file to store window information

SEE ALSO

toolplaces(1), xprop(1), xwininfo(1), xplaces(X)

BUGS

Xtoolplaces goes VERY out it's way to capture as many X applications as possible (including ill-behaved ones :-{). The most up-to-date information comes from WM_COMMAND if the application understands WM_SAVE_YOURSELF. Don't blame me if all the information isn't there for ill-behaved applications, I tried!

If the application doesn't put the icon position in WM_COMMAND or into window manager hints, there is no way to save that information.

For windows in which the geometry has to be gotten manually, xtoolplaces doesn't put on the command line whether to start iconic or not (even if the application is iconic at the time). This is because different applications take different arguments to start iconic (if they CAN be started iconic).

This program WILL NOT work with mwm (Motif Window Manager) and tvtwm (the virtual version of twm). It seems to work with all the others (olwm, olvwm, twm) from what I've been told (I only use olvwm myself).

AUTHOR

Adam Stein
800 Phillips Road, 129-71B
Webster, New York 14580
Internet: adam@iset.scan.mc.xerox.com
stein.wbst129@xerox.com