NAME

larswm – Tiling Window Manager for X

SYNOPSIS

larswm [**–display** *dpy*] [**–f** *file*] [**–defaults**] [**–v**]

DESCRIPTION

larswm is a tiling window manager for X11, based on 9wm by David Hogan. It provides virtual desktops, support for tiled and untiled windows, keyboard shortcuts and more, while using very little system resources. Please see http://larswm.fnurt.net for more information.

OPTIONS

The following command line options are supported:

–display *dpy*	lets you specify which display you want larswm to manage.
–f *file*	lets you specify a preferences file other than $HOME/.larswmrc to use for configuration.
–defaults	prints to standard out a complete resource file for larswm containing all options and their defaults.
–v	prints the current version on standard error, then exits.

FILES

$HOME/.larswmrc /etc/X11/larswmrc

these two files are looked for in the above order when larswm starts up.

The Idea

When I started working on larswm I had a few ideas that I think are important in a window manager. They are as follows:

1. The user should not have to spend alot of time arranging windows on the screen, leaving more time for the actual work that he or she is trying to accomplish.

2. Direct manipulation. Instead of accessing windows and desktops through representations of these, access them directly. That means: No icons when a window is iconified. No graphical map of the virtual desktops, larswm is fast enough that it is easier to browse the actual desktops instead, something that is very quick and easy to do, especially if you have a wheel mouse, and since the windows are tiled, you usually do not need much time looking at a desktop to know what is on it. And within one desktop, tiling is a good example of direct manipulation, a common way to deal with multiple windows on the screen is to overlap them and select which one you want my clicking on a representation in the form of a button on a task bar, however, larswm does not need that because the actual window, with enough content visible to know which one it is, is always visible.

3. No precision clicking. Use the edges of the monitor to make it easy to aim with the mouse pointer.

4. It must be predictable. If the user ever wonders how focus ended up on a window that he or she did not click on, something is wrong. larswm has alot of code that deals with figuring out where focus should go when different events occur.

5. It must have a solid feel to it. Part of this is speed, when you switch desktops or windows are moved around as the result of being tiled it must go quick, and part of it is predictability.

6. The window manager should not use more system resources (CPU time, virtual memory) than is

absolutely necessary for performing the task of managing windows. The GUI is minimal, providing borders around the windows, and a status bar at the bottom of the screen. It provides all feedback using plain text on the status bar.

7. Quality control. It has been tested with Electric Fence to catch memory access errors and profiled with gprof. So far I have been able to fix everything that I found with these tools, but I am no CS major and I am sure some things could be done better. Nevertheless, it works, and during normal circumstances it works using minimal system resources.

Terminology

Some of this is unique to larswm and some is borrowed from other GUI designs.

Screen

Physical monitor. By default larswm will use up to 4 monitors. This can be adjusted in the dat.h file.

Virtual Desktop

Each screen is logically divided into up to 16 workspaces. A window is usually only visible on one of these workspaces.

Subdesktop

Each virtual desktop is then logically divided between two subdesktops. First we have normal, untiled, windows. They are managed in a way similar to most other window managers. Second we the tiled subdesktop, where no window is allowed to overlap another. Switching between the tiled and the untiled set of windows is very fast with a hotkey or mouse click.

Track

On the tiled subdesktop, windows are laid out in one of two tracks. By default, the ratio is 60%/40% for left/right track. The ratio can be changed dynamically through hotkeys, and a different default can be set in the .larswmrc file. The left track always contains one window filling it from top to bottom while the right track contains any other windows, evenly sized. There are hotkeys to make any window in the right track expand and move to the left track while the window that was in the left track is shrunk and placed at the top of the right track.

Status Bar

The status bar fills two functions. First off, as the name implies, it shows status messages. This includes the window title of the focused window, the selected virtual desktop, flags that shows what modes are set on the current virtual desktop and a user defined message that is used by the sample clock app to show the current date and time. Second, it is a very big button that can be clicked to perform various functions. It is easy to target, since you just move the mouse down against the bottom of the screen, no need to precision aim for it.

Tool

A special kind of window that belongs to the untiled subdesktop, but is still tiled around. It is also visible on all virtual desktops. Used for things like clocks and load meters.

Select Zoom

Maximize a window so that the height/width ratio is the same as for a US letter sized paper, and centering it on the screen. Besides wanting to tile windows, this was the most common manual move/resize operation I used before it was automated.

Put Aside

All this means is that a window is moved mostly off screen, leaving just a portion visible. It is another way to do iconifi cation.

A Sample Session

The following assumes you are using the sample.xsession and sample.larswm confi g fi les. After logging in, you will see a mostly empty desktop. Along the bottom of the screen is the status bar, and in the lower right corner are three smaller windows (xload, xbiff and xclock) which in the default confi g are tagged as tool windows. What this means is they should be visible on all virtual desktops, and should not be covered by windows that are placed automatically on the tiled subdesktop.

Working with tiled windows

Now hit Shift-Control-Return. This will bring up a new terminal window. Notice how it occupies the left 60% of the screen. Type something in this window and hit Shift-Control-Return again. The fi rst window is moved to the right side and shrunk, while the new xterm occupies the bigger area to the left. Notice that the toolwindows (xload, xbiff and xclock) are not overlapped.

Hit Shift-Control-Return once more and notice how the two previous windows now both are on the right side and using half the space each.

Now hit Control-Button1 on the fi rst window you opened, it should be the xterm right above the tool windows on the right side. This will put that window in the left track, while putting the left track window at the top of the right track.

That is all there is to never having to manually line up your xterms again!

Working with untiled windows

For demonstration purposes, we can use xlogo. By default, the xlogo window class is not set to be tiled, so it will be opened on the untiled subdesktop, separate from all the xterms. In the focused xterm, type xlogo and hit Return. The xlogo window should appear in the middle of the screen above the tiled windows. If you need to switch between tiled and untiled windows, you can use Shift-Control-BackSpace. Hit that key twice and see how xlogo is fi rst hidden behind the tiled xterms and then brought to the front again. If you had more untiled windows open, they would all have been lowered and raised together.

Moving windows to predefined areas

You can move untiled windows to 9 different predefi ned locations on the screen with only one keystroke. With the xlogo window focused, hit Shift-Control-KP_6. The xlogo window should be against the right edge of the screen, halfway between the top and bottom, just like 6 on the numeric keypad is among the keys 1 through 9. Hit Shift-Control and each number on the numeric keypad to see the locations. When done, leave the window on the right edge in the middle (Shift-Control-KP_6).

Select zooming

With the xlogo window focused and off to the side of the screen, hit Shift-Control-space. Notice how it is centered and sized to make it easy to work with. When you hit Shift-Control-space again, it is returned to its former place and size. This can be used as an alternative to tiling, if you have some windows you would like to mostly manage yourself, but want a quick way to concentrate on one of them.

Set aside windows

Sometimes when working on the untiled desktop you just want to set a window aside for a moment. Hit Control-Alt-space to move a window sideways off the right edge of the screen, leaving just a small portion visible. Clicking on the visible part will put that window back where it was. If you hit Control-Alt-space with no untiled window focused, the last put aside window will be brought back. See the mouse reference section for more ways to manipulate windows in this fashion.

Using the mouse to move and resize windows

With the xlogo window unzoomed and focused, move the mouse down onto the status bar, anywhere will do, and hold down Shift-Alt-Button1. The mousepointer moved to the top left corner of the window and you are able to move the window around by moving the mouse. To place the window, let go of the mouse button. Move xlogo to somewhere near the top left of the screen.

Next, with the mouse on the status bar, hold down Shift-Control-Alt-Button1. The mouse pointer is moved to the bottom right corner of the xlogo window and when you move the mouse, you resize the window. To set the size, let go of the mouse button.

Moving a window between subdesktops

Make sure the xlogo window is focused and then hit Shift+Alt+space. The xlogo should now be tiled and focused in the left track. If you hit Shift+Alt+space once more, it becomes untiled again and the other tiled windows moves to fi ll the vacated spot in the left track.

Working with virtual desktops

So far, we have only used the fi rst desktop. The way to switch between desktops is easy. Shift-Control-Left and Shift-Control-Right on the keyboard will move you back and forth between the desktops. Hit Shift-Control-Right four times. Notice how the status bar label changes to show what desktop is active. By default there are four virtual desktops per screen, so you should be back to the fi rst desktop now.

If you instead use Shift-Alt-Left or Shift-Alt-Right you will move to the new virtual desktop while dragging the currently focused window with you.

Using the mouse wheel

With larswm, the mouse wheel can be used for quite a few operations. The most common is to switch virtual desktop. This can be done by moving the pointer to the status bar and rolling the wheel up and down.

You can also use the mouse wheel to move and resize windows quickly. With xlogo focused, move the pointer to the status bar and hold down Shift-Alt while rolling the wheel to move the window sideways. If you hold down Shift-Control-Alt you move the window up and down, and with Shift-Control you can grow and shrink the window.

If you have more than one window open, hold down Shift while rolling the wheel down to cycle focus through each window. To try it now, shift to the tiled subdesktop by clicking on any of the tiled xterms then move the mouse pointer to the status bar, hold down shift and roll the wheel.

This is not exactly intuitive, but very fast once you learn the combinations of Shift, Control and Alt keys to do these things.

Status Bar Menu System

The status bar is used for a simple menu system. With the pointer on the status bar, hold down Control and click Button1 and Button3, or roll the wheel up and down, to scroll between the menu entries. Control-Button2 selects the current menu entry, usually toggling a desktop or window setting.

If you instead of Control hold down Alt, you can scroll through the list of iconifi ed windows, and Alt-Button2 uniconifi es the selected window.

Things You Can Configure

You can affect alot (but not all!) of the behaviour of larswm by making your own .larswmrc fi le and putting it in your home directory.

Default setup

Included in the distribution fi les is a fi le called sample.larswmrc that you can use as a starting point for your own confi g.

Complete list of config keywords

Another way to get a default .larswmrc is to run larswm with the -defaults flag. It will print out all the possible options with their defaults fi lled in, all you have to do is uncomment and edit any lines you wish to change. Rather than reproduce that here, you can create your own up to date version by running the command larswm -defaults and piping it to a fi le.

Keyboard commands for all windows and subdesktops

Shift-Control	Prior	Prev Screen
Shift-Control	Next	Next Screen
Shift-Control	Left	Prev Desktop
Shift-Alt	Left	Prev Desktop Drag Focused Window
Shift-Control	Right	Next Desktop
Shift-Alt	Right	Next Desktop Drag Focused Window
Shift-Control	F1-F12	Jump to desktop 1-12
Shift-Control	BackSpace	Toggle Subdesktop
Shift-Control	Up	Prev Window
Shift-Control	Down	Next Window
Control-Alt	z	Hide Window
Control-Alt	x	Unhide last hidden window
Control-Alt	w	Close Window
Shift-Alt	space	Move window to other subdesktop

Keyboard commands for untiled windows

Shift-Control	space	Select Zoom
Control-Alt	space	Put/restore aside window
Shift-Control	0	Put aside all other windows
Shift-Control	KP_Home	Move window to top left
Shift-Control	KP_Up	Move window to top center
Shift-Control	KP_Prior	Move window to top right
Shift-Control	KP_Left	Move window to left center
Shift-Control	KP_Begin	Center window
Shift-Control	KP_Right	Move window to right center
Shift-Control	KP_End	Move window to bottom left
Shift-Control	KP_Down	Move window to bottom center
Shift-Control	KP_Next	Move window to bottom right
Shift-Alt	KP_Up	Move window up
Shift-Alt	KP_Down	Move window down
Shift-Alt	KP_Left	Move window left
Shift-Alt	KP_Right	Move window right
Shift-Control-Alt	KP_Up	Grow window vertically
Shift-Control-Alt	KP_Down	Shrink window vertically
Shift-Control-Alt	KP_Left	Shrink window horizontally
Shift-Control-Alt	KP_Right	Grow window horizontally
Shift-Control-Alt	KP_Home	Grow window both ways
Shift-Control-Alt	KP_End	Shrink window both ways
Shift-Alt	KP_Insert	Toggle move/resize increment
Shift-Alt	KP_Home	Maximize window vertically
Shift-Alt	KP_End	Maximize window horizontally
Shift-Alt	KP_Begin	Maximize window
Shift-Control-Alt	KP_Begin	Full screen

Keyboard commands for tiled windows

Shift-Control-Alt	KP_Up	Maximize left track
Shift-Control-Alt	KP_Down	Restore left track
Shift-Control-Alt	KP_Left	Shrink left track
Shift-Control-Alt	KP_Right	Grow left track
Shift-Alt	KP_Begin	Move window to/from left track
Control-Alt	space	Restore aside window

Mouse buttons on the status bar

Modifi er	Button1	Button2	Button3
None	Prev Desktop	Subdesktop	Next Desktop
Control	Prev Menu	Select Menu	Next Menu
Alt	Prev Hidden	Hide/Unhide	Next Hidden
Shift-Control	Shrink	Select Zoom	Grow
Shift-Alt	Move Window		
Shift-Control-Alt	Resize Window	Lower Window	Maximize Window
Control-Alt	Restore Aside	Put/Restore Aside	Put Aside

Mouse wheel on the status bar

Modifier	Roll Up	Roll Down
None	Prev Desktop	Next Desktop
Shift	Prev Window	Next Window
Control	Prev Menu	Next Menu
Alt	Prev Hidden	Next Hidden
Shift-Control	Shrink	Grow
Shift-Alt	Move Up	Move Down
Shift-Control-Alt	Move Left	Move Right
Control-Alt	Restore Aside	Put Aside

Clicking on unfocused windows

Button1 is used to focus windows.

If skip_focus is on (this is the default) you can force a tiled window into the left track by Control-Button1 clicking.

If skip_focus is off, then Control-Button1 clicking will prevent moving the newly focused window to the left track.

If you Control-Button1 click on a window that is not on the active subdesktop, you will switch focus without also switching subdesktop.

If you Control-Button1 click on an aside window, that window will be put back while all other untiled windows are put aside.

Indexed resources

In the sample.larswmrc, all the entries that end in a number are indexed resources. The number can be between 0 and 63, and must be unique within each resource type.

Multi-head

When specifying an option that only affects one of the virtual desktops, you can also limit it to a specific monitor. For instance, larswm.0.0.dtname: One would name the first virtual desktop on the first monitor only. Due to precedence, you can set an option for all desktops one way, and then specific desktops another. The ways to specify screens and desktops are, listed in order of precedence:

Format	Affected desktops/screens
larswm.S.D.resource	Only desktop D on screen S
larswm.S.?.resource	All desktops on screen S
larswm.?.D.resource	Only desktop D on all screens.
larswm.?.?.resource	All desktops on all screens.
larswm*resource	All desktops on all screens.

As an example, if you want all desktops except the first on the second monitor to not resize windows that are tiled, you would put the following lines in your .larswmrc:

```
larswm*tile_resize: false

larswm.1.0.tile_resize: true
```

The fi rst line is needed because the default is to resize windows.

The Status Bar

The thin window across the bottom of the screen is the status bar. On the left is the window title of the focused window, and on the right is a status area.

Mode flags

Between the virtual desktop label and the user defi ned message (date/time in the default setup) is a block of flags in upper case [TCRSBH]. They indicate the following:

T **Subdesktop**

T for tiled or U for untiled.

C **Clickthru**

Whether clicks on an unfocused window should focus and pass the click on to the client, or just focus and discard the click. C means pass to client.

R **Tile Resizing**

Whether tiled windows are resized or not. R means they are resized. If not, they are just stacked in the right track, each having the top left corner visible.

S **Skip Focus**

Whether giving focus to a window in the right track makes it jump to the left track. S means no, you have to Control-Button1 click it to make it jump. If off, you can Control-Button1 click to make a window stay in place while getting focus.

B **Big Move/Resize**

Whether the keyboard and mouse wheel commands to move and resize windows should work a pixel at a time, or 5% of the screen width/height at a time.

H **Hidden Windows**

Whether any windows are hidden (iconifi ed) on any screens. It is there so you will not forget about them, since larswm has no visual representation of iconifi ed windows.

Window mode flags

The status bar will also show the class of the focused window and instance in the format (Class˜Instance). Just to the right of that is another set of flags in lower case [nftsz] that mean:

n **Notile**

Window belongs to the untiled subdesktop.

f **Floating**

Window always stays on top of non-floating windows.

t **Tool**

Window is tiled around, not over.

s **Sticky**

Window is visible on all virtual desktops.

z **Zoomed**

Window is in one of the zoomed states.

User defi ned message

If you want, you can put any message you like in the right most portion of the status bar. This can be used for things like clocks, email notifi ers, stock tickers etc. Included in the distribution is a small example of a clock called larsclock. It is started by the default session script.

If you do not run larsclock, you can set the message from any shell script with the command larsremote message TEXT, which would show TEXT on the status bar.

Window Types

Here is a more detailed look at the different types larswm can associate with a window.

Transient windows

Windows that have the WM_TRANSIENT_FOR property set are put on the untiled subdesktop even if the window class has been specifi ed as tiled.

When a transient window fi rst gets mapped, it will always show up, no matter what desktop you are viewing. The transient window will be assigned to the desktop where the parent window is though, so as soon as you change desktop, it will only show up on the virtual desktop the parent belongs to.

Untiled windows

Any window that is not tiled is assigned to the untiled subdesktop. These windows are completely separate from the tiled subdesktop. The untiled subdesktop behaves more or less like any traditional window manager, but adds a few keyboard shortcuts for moving windows to different parts of the screen or zooming them in both dimensions.

larswm will not tile a window unless its class/instance can be found in the larswm.dotileclass list.

The way to move between your tiled and untiled desktop is through a keyboard shortcut, which by default is Shift-Control-BackSpace

When a window is mapped on the untiled subdesktop for the fi rst time, it is centered on the screen, unless it has had its desired position set in the hints.

Floating windows

Floating windows will always stay on top of other non-fbating windows, regardless of which subdesktop is active. This can be used for small windows, like xcalc, that you always want accessible. If clickthru is on, clicks will be passed on to fbating windows regardless of which subdesktop was active before the click.

Sticky windows

Sticky windows are windows of a class and instance that matches an entry in the larswm.stickyclass list. They are always visible no matter which virtual desktop you are on.

Tools

Tools are windows you always want visible. Examples of tools are xclock, xload and other small informational windows. When a window has been identifi ed as a tool window, it will be assigned to the untiled desktop, it will be sticky and the tiled windows will not cover it. Also, if clickthru is on, any clicks on tool windows will be passed on to the client, regardless of which subdesktop was active before the click. Tools will never end up focused unless you click on them.

Internals

A little more detail about how larswm does things.

Every time a window is mapped or unmapped, or (if skip_focus is off) you focus a new window on the tiled subdesktop by clicking on it the following process is executed:

1. The width of the two tracks are calculated. This is by default set up so the left track uses 60% of the screen width, but it can be confi gured in the

2. It calculates how many windows should be in each track, one in the left track and the rest in the right.

3. The height of each window in each track is calculated.

4. It goes through the list of all windows and places them in the designated spot.

5. After this it returns to the event loop.

That is pretty much it. If skip_focus is off (default is on), when you click on an inactive window in the right track, it will pop over into the left track, and the window that were there will pop over to the top of the right track.

At all times, the right track will contain tiled windows in the order that they were in the left track, starting with most recent.

Associating windows with a virtual desktop

You can associate certain window classes/instances to a certain virtual desktop number by adding resources to your .larswmrc.

An example of how to associate all Netscape windows with the second virtual desktop:

larswm.dtclass.0: Netscape

larswm.dtnum.0: 1

When any Netscape window opens, larswm will fi rst switch to the specifi ed desktop, then map the window.

Hidden windows

larswm iconifi es windows by hiding them. To unhide a hidden window, you must fi rst locate its label using the status bar. Please see the section about using menus on the status bar for information on how this works.

You can also unhide the last hidden window with one keystroke, Control-Alt-x.

Focus handling

To change focus to a window, you just left click on it, and if skip_focus is off that window will pop into the left track, and the window currently there will pop over to the top of the right track.

If you have it set up to automatically put the focused window in the left track, you might still sometimes want to type something into one of the tiled windows without rearranging all the windows, and to do that you Control-Button1 click on a window. Focus will change, but all windows will stay where they are. This will also prevent any transient windows owned by the application from immediately be raised and get focus.

If you have it in the default mode, where selecting a new window to have focus does not make it pop into the left track you can do that by Control-Button1 clicking it.

Clickthru

When you click on a window to give it focus, that click is also passed on to the client. This makes it easier to work with a multi-window application. You can turn this feature off, in which case the click used to focus a window is never seen by the application.

Multi-Head

When a window gets mapped, for whatever reason, it is usually given focus. But if the MapRequest event occurs on a different screen than the one you are working on, things could get confusing. I solved this by adding the following rules to how focus is assigned when windows open and close:

1. If the currently focused window is on the same screen, then the new window will get focus.

2. If the currently focused window is on a different screen, the new window will get focus only if the mouse pointer is on the same screen as the new window.

3. If no window is focused, a newly mapped window will get focus only if the mouse pointer is on the same screen as the new window.

4. If the focused window closes, focus will revert to another window, usually the window that was focused before the one that just closed. Focus will revert only if the mouse pointer is on the same screen as the window that closed.

I believe these rules will help eliminate most cases of having your keystrokes go to a window you did not intend it to.

Atoms

The following atoms are created and monitored by larswm and can be used in third party tools:

LARSWM_EXIT

Tells larswm to exit.

LARSWM_RESTART

Tells larswm to restart, reloading the .larswmrc fi le.

LARSWM_BARTEXT

When this is updated, the content of this atom is shown as text on the status bar until the atom is updated again. See larsclock and larsremote for examples of how to use this.

Compatibility

Terminal Windows

xterm windows might look like they are not redrawing correctly, but they are in fact. It is up to the program running inside it to detect window size changes and redraw the screen as necessary. One way to deal with this is to run screen inside the xterm, as it will handle resizing and redrawing correctly.

On some, mostly older, systems you need to make terminal windows know that they have been resized when they are first opened and tiled.

Put this line in your .cshrc or .bashrc:

```
eval `resize`
```

It will ensure that it is set to the correct number of rows and columns when it is opened.

Size Hints

The PResizeInc and PMaxSize hints are honored. PMinSize is not, as it might have problems fitting all the windows in when tiling them.

Resizing Windows

For windows that do not cope well with being resized, you can disable automatic resizing on a particular desktop, making the windows in the right track be stacked on top of eachother instead. This can also be used for windows that resize slowly, like Netscape, to speed things up considerably.

Multi-Head Displays

It does work well with multi-head displays. One of my development systems is a dual-head Sun running Solaris 9, so this functionality has been tested quite alot. It currently behaves in a predictable manner when you switch focus between screens, move between desktops etc.

GTK Applications

Some GTK apps do not set WM_TRANSIENT_FOR correctly on their dialog boxes, and they also sometimes have a different class string on those subwindows, making things confused. The best way to deal with those kinds of problems is to specify both the class and instance of the top level windows you do want tiled. In some cases, this still does not work, like in some builds of Mozilla where every window it creates has class/instance Mozilla-bin/mozilla-bin.

Standards Compliance

At this time there is no GNOME/KDE/Motif/whatever compatibility. It follows the ICCCM as much as it can while still providing all the automatic functionality that it does.

LICENSE

Many thanks to David Hogan for writing 9wm and releasing it under such a license that I could use it as a base to build larswm on.

Here is the original license for 9wm:

SEE ALSO

larsclock(1x), larsmenu(1x), larsremote(1x)

AUTHORS

larswm was created by Lars Bernhardsson <lab@fnurt.net> by building on 9wm by David Hogan.