Subject: Windows drives vs POSIX mounts
Posted by guido on Fri, 28 Apr 2006 13:36:26 GMT
View Forum Message <> Reply to Message

I tried to port FileSel to POSIX.
Unfortunately it makes assumptions which work for the MS-DOS drive letter system only.
In the POSIX world "drives" are mounted into /mnt and /media or /Volumes in the case of MacOS X.
Only the boot disk is mounted directly into the root (/).

I hacked FileSel to have it work better with modern Linux systems, conforming to POSIX FHS 2.3:

```
Image PosixGetDriveImage(String dir)
{
 if(dir.Find("cdrom") == 0 || dir.Find("cdrecorder") == 0)
  return CtrlImg::CdRom();
 if(dir.Find("floppy") == 0 || dir.Find("zip") == 0)
  return CtrlImg::Diskette();
 // if (??) return CtrlImg::Computer();

 return CtrlImg::Hd();
}


bool Load(FileList& list, const String& dir, const char *patterns, bool dirs,
          Callback3<bool, const String&, Image&> WhenIcon, FileSystemInfo& filesystem)
{
 if(dir.IsEmpty()) {
  Array<FileSystemInfo::FileInfo> root = filesystem.Find(Null);
  for(int i = 0; i < root.GetCount(); i++)
   list.Add(root[i].filename, GetDriveImage(root[i].root_style),
    Arial(FNTSIZE).Bold(), SBlack, true, -1, Null, SCyan,
    root[i].root_desc, Arial(FNTSIZE));
 }
 else {
  Array<FileSystemInfo::FileInfo> ffi =
   filesystem.Find(AppendFileName(dir, filesystem.IsWin32() ? "*.*" : "*"));
  if(ffi.IsEmpty())
   return false;
#ifdef PLATFORM_POSIX
  bool isdrive = dir == "/media" || dir == "/mnt";
#endif
  for(int t = 0; t < ffi.GetCount(); t++) {
   const FileSystemInfo::FileInfo& fi = ffi[t];
#ifdef PLATFORM_POSIX
   Image img = fi.is_directory ?
    (isdrive ? PosixGetDriveImage(fi.filename) : CtrlImg::Dir()) :
```

```
    CtrlImg::File();
#else
   Image img = fi.is_directory ? CtrlImg::Dir() : CtrlImg::File();
#endif
   WhenIcon(fi.is_directory, fi.filename, img);
   bool nd = dirs && !fi.is_directory;
   if(fi.filename != "." && fi.filename != ".." != 0 && fi.filename.Find('.') != 0 &&
      (fi.is_directory || PatternMatchMulti(patterns, fi.filename)))
    list.Add(fi.filename, img,
       fi.is_directory ? Arial(FNTSIZE).Bold() : Arial(FNTSIZE),
       nd ? SGray : SBlack, fi.is_directory, fi.is_directory ? -1 : (int)fi.length,
       Null, nd ? SGray : fi.is_directory ? SBlack : SLtBlue);
  }
 }
 return true;
}
```

(I made it hide dotfiles also - maybe open/save dialog needs a show-hidden toggle for POSIX)

Still, the drives DropList doesn't get populated.
But this code snipped as meant as demonstration only- it's really a hack. FileSel needs to be rewritten, allowing for the "conceptual root" of POSIX systems, no longer assuming all drives are to be found in "/" flat.

Sorry if code snipped comes out distorted!
How do I do that properly?

Subject: Re: Windows drives vs POSIX mounts
Posted by mirek on Fri, 28 Apr 2006 14:27:27 GMT
View Forum Message <> Reply to Message

guido wrote on Fri, 28 April 2006 09:36I tried to port FileSel to POSIX.
Unfortunately it makes assumptions which work for the MS-DOS drive letter system only.
In the POSIX world "drives" are mounted into /mnt and /media or /Volumes in the case of MacOS X.
Only the boot disk is mounted directly into the root (/).

I hacked FileSel to have it work better with modern Linux systems, conforming to POSIX FHS 2.3:

```
Image PosixGetDriveImage(String dir)
{
 if(dir.Find("cdrom") == 0 || dir.Find("cdrecorder") == 0)
  return CtrlImg::CdRom();
 if(dir.Find("floppy") == 0 || dir.Find("zip") == 0)
  return CtrlImg::Diskette();
```

```
 // if (??) return CtrlImg::Computer();

 return CtrlImg::Hd();
}


bool Load(FileList& list, const String& dir, const char *patterns, bool dirs,
          Callback3<bool, const String&, Image&> WhenIcon, FileSystemInfo& filesystem)
{
 if(dir.IsEmpty()) {
  Array<FileSystemInfo::FileInfo> root = filesystem.Find(Null);
  for(int i = 0; i < root.GetCount(); i++)
   list.Add(root[i].filename, GetDriveImage(root[i].root_style),
    Arial(FNTSIZE).Bold(), SBlack, true, -1, Null, SCyan,
    root[i].root_desc, Arial(FNTSIZE));
 }
 else {
  Array<FileSystemInfo::FileInfo> ffi =
   filesystem.Find(AppendFileName(dir, filesystem.IsWin32() ? "*.*" : "*"));
  if(ffi.IsEmpty())
   return false;
#ifdef PLATFORM_POSIX
  bool isdrive = dir == "/media" || dir == "/mnt";
#endif
  for(int t = 0; t < ffi.GetCount(); t++) {
   const FileSystemInfo::FileInfo& fi = ffi[t];
#ifdef PLATFORM_POSIX
   Image img = fi.is_directory ?
    (isdrive ? PosixGetDriveImage(fi.filename) : CtrlImg::Dir()) :
    CtrlImg::File();
#else
   Image img = fi.is_directory ? CtrlImg::Dir() : CtrlImg::File();
#endif
   WhenIcon(fi.is_directory, fi.filename, img);
   bool nd = dirs && !fi.is_directory;
   if(fi.filename != "." && fi.filename != ".." != 0 && fi.filename.Find('.') != 0 &&
      (fi.is_directory || PatternMatchMulti(patterns, fi.filename)))
    list.Add(fi.filename, img,
       fi.is_directory ? Arial(FNTSIZE).Bold() : Arial(FNTSIZE),
       nd ? SGray : SBlack, fi.is_directory, fi.is_directory ? -1 : (int)fi.length,
       Null, nd ? SGray : fi.is_directory ? SBlack : SLtBlue);
  }
 }
 return true;
}
```

(I made it hide dotfiles also - maybe open/save dialog needs a show-hidden toggle for POSIX)

Still, the drives DropList doesn't get populated.
But this code snipped as meant as demonstration only- it's really a hack. FileSel needs to be rewritten, allowing for the "conceptual root" of POSIX systems, no longer assuming all drives are to be found in "/" flat.

Sorry if code snipped comes out distorted!
How do I do that properly?

Well, I guess, "porting" is quite strong word here - right now I am writing this from ubuntu, while developing using TheIDE - no problems with fileselector

However, adding some common root pathes to the fileselector droplist looks like a good idea to me. But I guess it should be done more carefuly - maybe depending just on names is not a good idea.

Concerning .dotfiles, yes, I guess, there definitely should be an option.

Mirek

Subject: Re: Windows drives vs POSIX mounts
Posted by forlano on Fri, 28 Apr 2006 19:12:48 GMT
View Forum Message <> Reply to Message

guido wrote on Fri, 28 April 2006 15:36
Sorry if code snipped comes out distorted!
How do I do that properly?

You should quote it with the tag

[code]
.... your code ....
[/code]

but withou blank space between the first '[' and 'code'. Without it in fact you obtain:

.... your code ....

Luigi

Subject: Re: Windows drives vs POSIX mounts

Posted by guido on Fri, 28 Apr 2006 19:43:26 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 28 April 2006 16:27
Well, I guess, "porting" is quite strong word here - right now I am writing this from ubuntu, while developing using TheIDE - no problems with fileselector

However, adding some common root pathes to the fileselector droplist looks like a good idea to me. But I guess it should be done more carefuly - maybe depending just on names is not a good idea.

Concerning .dotfiles, yes, I guess, there definitely should be an option.

Mirek

Sorry, current file-selector doesn't support removable media on Linux. Even if the users knows about /media (which she shouldn't need to), FileSel still only show mounts as regular folders.
If this miraculously isn't true for Ubuntu I'd like to know how so.

And the names below /media are *POSIX STANDARD* - I haven't picked them arbitrarily or because they happen to be true on my current favourite distro du jour.
Obviously this isn't a 100% solution. But POSIX has no API for distinguishing (physical) devices. You can figure out, if a directory is a mount point in a portable way, but that's it.
So you either need to add a libsysfs plugin for Linux and something else for FreeBSD and something else for MacOS X later on - or open the libhal/dbus can of worms, which for various reason you really don't want to. Ask, if you don't know what I'm meaning. libhal maybe ok in a couple of years, now it isn't for upp IMHO.

As the solution I presented doesn't break FileSel for anybody, while making it work better for modern standard compliant Linuxes, I suppose, I don't see why you would pass it off. View it as a stop-gap, if you will.

I understand that upp wasn't meant to be portable initially. Even though upp almost works on Linux now, the code still is strongly biased towards win32 in places. FileSel down to FileSystemInfo especially lacks many features on POSIX by design.
I don't see how abstracting root and drive handling from the currently flat drive(letter) system can be avoided, if you are serious about POSIX implementations. On MacOS X /Volumes is a hidden directory, so users there might not even be able to access this directory from upp's file-selector, once ported. I'm not sure so on what level hiding is done - filesystem, libc, Carbon/Cocoa, Finder, so maybe /Volumes indeed would show up in upp apps - but Mac users are notorious for not tolerating such alien means.

Subject: Re: Windows drives vs POSIX mounts
Posted by mirek on Sat, 29 Apr 2006 05:52:35 GMT
View Forum Message <> Reply to Message

guido wrote on Fri, 28 April 2006 15:43
Sorry, current file-selector doesn't support removable media on Linux. Even if the users knows about /media (which she shouldn't need to), FileSel still only show mounts as regular folders. If this miraculously isn't true for Ubuntu I'd like to know how so.

I am sorry for arguing, perhaps I just do not understand the issue right, but what is wrong with

"FileSel still only show mounts as regular folders"

? I always thought that this is a big advantage of unified posix filesystem?

Quote:
And the names below /media are *POSIX STANDARD* - I haven't picked them arbitrarily or because they happen to be true on my current favourite distro du jour.

OK, I am convinced. Anyway, have I got it right that your proposal in fact covers two areas:

- directory icons
- directory history droplist initial content

?

Mirek

Subject: Re: Windows drives vs POSIX mounts
Posted by guido on Sat, 29 Apr 2006 12:09:02 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 29 April 2006 07:52

I am sorry for arguing, perhaps I just do not understand the issue right, but what is wrong with

"FileSel still only show mounts as regular folders"

? I always thought that this is a big advantage of unified posix filesystem?

OK, I am convinced. Anyway, have I got it right that your proposal in fact covers two areas:

- directory icons
- directory history droplist initial content

?

Mirek

/media is for user-mountable devices.

Unified filesystem is about mounting e.g.
/home /var /boot /usr /srv
onto seperate local hd-partitions or a remote resource even - seemlessly, flexible and predictably. This makes it easy for the admin in enterprise environments to partition a system to his liking without the client user noticing a difference.
Also there are other mounts like /dev /proc /sys, possibly /tmp, which like the above shouldn't prominently show up in a file-selector, because they are of no interest to anybody than the system administrator, developers (sometimes) - or your distributor for a home box.

But /media is special. That's the place for removable/hotplugged storage. Or non-system internal disk-partitions, like a partition filled with multimedia files. It should be easily discoverable.
Those places show up on the Nautilus and KDE desktops also, if the distributor has done his job well, at least.
KDE and Gtk file-selectors support even more special locations in the filesystem like $HOME/Desktop. I think this is configurable to some extend though - they use a bookmark system here I think. Probably fairly easy to have FileSel read out the KDE file-selector "bookmarks" - but that's no "must have" I think.

POSIX filesystems are just to arcane and cluttered, due to the convention of installing by filetype (I like to call this "splatter install"), to have non-experts freely navigate outside $HOME, so Linux desktops, notably Nautilus and Konqueror give it a friendlier face with an alternate view. This abstraction above the filesystem has the additional advantage of being easily localizable (avoids the problems MS created for installers by localising "Program Files" directly on the filesystem level).
Gnome/KDE use their VFS layers for this purpose (gnome-vfs, kio respectivly) - but for this purpose alone you don't need a full-blown userland VFS, as you can see. There is effort underway to have - shared by KDE and Gnome - "desktop VFS". If it takes off, upp could use that later on, but who knows if it does take off...
MacOS X can afford to go even further by hiding the POSIX tree alltogether, as there the POSIX tree doesn't hold GUI relevant files like icons, wallpapers, fonts.

Btw:
In my code I added /mnt only for legacy purposes, as it's now supposed to be a system administrator's playground only. It's empty on a home maschine on recent Linuxes normally now. On legacy systems it still has the purpose /media has now.
As the droplist should contain the mounts below /mnt and /media - not those directories as such, it doesn't hurt to have /mnt checked for drives as well.

Yes, drive images, droplist with drives and dotfile support, is what I want. Gnome and KDE users will expect it and else will few upp as inferior.
Personally I think you should make the POSIX file-selector look like KDE's, that would be a strong plus, put it even ahead of plain Qt, as Qt suffers from the same win32 default look&feel problem as upp (and most other toolkits, except wx). toolkit-theme-engines/skins isn't the whole story - which leads me to an additional request:

Would be nice if upp supported XDG compliant icon themes, and apps would use built-in icons only as fallback - and I want a pony

Would be cool if unmounted drives could be mounted from the fileselector - otherwise there is no point showing them at all.
Omitting unmounted drives or mount'em on double-click - not sure what's better honestly.
