Universal themes: a proposal

Judging by the number of themes submitted to kde-look.org, art.gnome.org, xfce-look.org and gnome-look.org, customising the look and feel of a free desktop is important to the users.

Unfortunately, the situation is far from ideal:

- A lot of themes are simply packaged in a tarball (.tar.gz), leaving the user to guess how to install and activate them. Opening these themes in the default application rarely helps.
- While appearance configuration utilities know how to handle some tarballs, they fail miserably when a theme has been archived and compressed in a unexpected way. Theme designers often choose to do so anyway, issuing instructions in README files. There is no specification on how a lot of themes should be packaged.
- Themes cannot be previewed in a file browser by their thumbnails.
- Themes, of different types, cannot be linked together or easily combined to form one complete look.

Attempts have been made to fix some of these problems, especially in KDE. For example, KDE includes “theme-manager themes” which are able to link to several different types of themes, as well as include a wallpaper and a colour scheme. This type of theme seems to have been deprecated in KDE 4, however.

This proposal suggests a new file type: **a universal theme (utheme).**

Use-case scenario: Jack visits his favourite website for customising his desktop. He finds a GTK theme that he likes, clicks on the link and downloads it to his desktop. The file has a handy little thumbnail, allowing Jack to preview the theme before installing it and giving him confidence that the file is not badly packaged. He double-clicks on it. A program automatically launches, installs it and activates it for him.

Use-case scenario: Sandra is browsing her favourite website, and is overwhelmed by the number of different types of themes she can install. Fortunately, she sees a category called “complete skins”. She browses skins by colours, and downloads a green one to her desktop. The file has a handy little thumbnail. She double-clicks on it, and a program automatically launches, installs it and activates it for her. Her entire desktop has changed looks, including her wallpaper, window decorators, widgets, log-in screen and boot-up screen.

Use-case scenario: Jessica is a theme designer. She is keen on creating coherent themes that are easily installable. She packages the themes in the utheme format, and she creates a “meta-skin” utheme that links them all together. She uploads these files to kde-look.org. She blogs about her latest creation, linking to the meta-skin utheme. Users of KDE can download and install this meta-skin easily. Behind the scenes, GHNS retrieves all the linked themes automatically.

Use-case scenario: Songbird developers have chosen to use their own theme format to customise the look of their program. However, they want to make this new theme format accessible to a maximum number of people. They choose to support utheme, meaning systems automatically know how to categorise and thumbnail it. Songbird themes can be included in skins as well.

This proposal only a specification of this format. It does not address how the programs should install and activate files of this format, and it does not address how websites should distribute them. Ideas are given at the end.

		Notes/Questions
Format	gzipped archive (.tar.gz)	Could other compressed archives be supported easily?
Extension	`.utheme`	The extensions .theme and .thm have been taken already. A unique extension will make mime-type detection simple.
Mime-type	`application/x-utheme`	
Magic number	*undecided, not needed?*	

The contents of this compressed archive will be:

File	**Description**	**Notes**
`preview.png`	A small thumbnail of the theme. Required.	
`utheme.xml`	An XML document containing information about the theme. Required.	XML is a well defined mark-up language, easily parsed by most APIs. Alternatives to consider are INI files, or a custom configuration file.
`COPYING`	A text file containing copyright information about the theme. Optional. If not provided, information about copyright is assumed to be in utheme.xml	
`data/`	A directory which contains any files used by the theme. Required (even if empty)	This directory could be eliminated, putting the needed files directly in the root of the archive. In addition, preview.png and utheme.xml could be in their own directory.

A theme packaged by utheme should be installable as-is on the intended target destination, it should not have to compiled for different architectures or systems. Themes that need to be compiled (like Usplash themes, or KDE's styles, or screensavers) are not supported by utheme, they should be packaged in the way binaries are usually packaged on the system.

The utheme.xml should contain the following information[1]:

- The utheme specification version number it complies to.
- The theme's type, often referring to the application that uses it, in lower-case, with no white-space. One theme type should have just one unique string by which it is defined. For example: gtk2, metacity, dekorator, firefox, songbird, plymouth, etc.
- The theme's unique code name, all in lower-case, with no white-space. Dashes are preferred to underscore characters to separate words. No other theme of the same type (unless it is a revision of the same theme) should share the code name. However, themes of different types can share the same name, for example, both a Metacity theme and a GTK2 theme can have

1Because this a draft open to discussion, how this information should be included and what XML should be used has not been specified.

the same code name `human`.

- The theme's name as read by humans. Translations of this name into different languages should be supported.
- The theme's description as read by humans. Translations of this name into different languages should be supported.
- The authors' names. Contact details (email address) may be included.
- Copyright information (who holds the copyright rights, and from what year).

utheme.xml can optionally contain the following information:

- A URL to the copyright license. A `COPYING` file may be included in the compressed archive as well. If the license referred to by the URL differs from the license in `COPYING`, both licenses apply. [Is this legally acceptable?] This field is highly recommended, as it enables categorising uthemes by license.
- The theme's version number. Version numbers should not be considered decimal numbers. Rather, they are similar to Debian's version numbers, in that they are separated by dots and leading zeros have no significance. (eg: 4.04 > 4.1) Pre-release version numbers are handled by trailing ~codename, again, similarly to Debian's version system.
- Primary and secondary colours that the theme uses can be specified, from the following list. Only the names should be used, not the HTML notation, as the purpose of the this information is to enable easy categorisation of the themes by colour. Colours to which the theme is closest to should be chosen. (Inspired by Opera skins)

- Dependencies on other themes can be included, as specified by their code names and their types. They should only be included if the theme cannot be installed or activated correctly without them (eg: an icon theme that inherits from another icon theme). An optional URL to the dependency's online location can be specified, however, a utheme capable program can override it if it can find the theme from another source (eg: GHNS)
- Dependencies on a program or a package can be specified. They should only be included if

the theme cannot be installed or activated correctly without them. This program or package should not be required for all themes of its type (eg: don't include a GTK dependency for GTK themes), but for it specifically (eg: do include dependencies on GTK2 engines). Each dependency can be specified by its PackageKit package name, by its package name per distribution, or by a URL from which an installer can be downloaded (eg: for Windows).

- Additional information specific to the theme type can be added as well.

This specification describes two special theme type: skins, and meta-skins. A **skin** is a collection of various themes of different types. They allow users to combine several themes into one coherent whole. Any number of themes can be included, but no more than one of each type, unless one is a dependency of another. Theses themes are included in the data/ directory of the compressed archive, and the file named `utheme.xml` lists them in the section specific to skins.

A **meta-skin** is not a theme in itself, rather, it links to other themes by their code names and types. They allow users to collect several themes into one coherent whole. Any number of themes can be linked to, but no more than one of each type. Meta-skins can optionally point to URLs where the uthemes can be downloaded, but this information can be ignored by the program. A meta-skin is to a utheme what a meta-package is to a package in APT. It is useful in spaces where repositories are used, like GHNS. Meta-skins are still uthemes, but they have empty data/ directories.

List of themes used in GNOME and KDE:

Theme type	**Current packaging**	**Suggested unique theme type string for utheme.xml**
Desktop background	Just an image file, sometimes a .xml file with images	`wallpaper`
Metacity border theme	`.tar.gz`	`metacity`
Xfce's border theme	`.tar.gz`	`xfwm4`
GTK2 controls theme	`.tar.gz`	`gtk2`
Icons theme	`.tar.gz`	`freedesktop-icons`
X11 mouse cursors theme	`.tar.gz`	`x11-cursors`
GRUB bootsplash	`.xpm` image file	`grub`
GDM login screen	`.tar.gz`	`gdm`
Plasma theme	`.tar.gz`	`plasma`
Colour scheme	`.kcsrc`	`kde-color-scheme`
KDE splash screen	`.tar.gz`	`kde-splash-screen`
GNOME splash screen	Just an image file	`gnome-splash-screen`
deKorator theme	`.tar.gz`	`dekorator`
Aurorae theme	`.tar.gz`	`aurorae`
Emeral theme	`.emerald`	`emerald`
Emoticon themes for KDE	`.tar.gz`	`kde-emoticons`
VLC theme	`.vlt`	`vlc`
Amarok theme	`.tar.gz`	`amarok`

And so on.

Discussion of alternative ideas:

One idea to solve the ambiguity of themes packaged as .tar.gz tarballs is to assign every theme type a new mime-type and extension. For example, .gtk2 would be assigned application/x-gtk2-theme and .metacity application/x-metacity-theme. Each type of theme would have its own thumbnailer, its own associated program and its own way to package the theme and its meta-data.

However, this idea makes it hard to create and maintain skins that link different kinds of themes together. Meta-data is not in one uniform format, which is handy when categorising and searching for themes, say for colour, license or author, on a server or on a desktop.

Another idea is to create “sub-mime-types” of utheme for each type of theme. For example, take a .metacity (application/x-metacity-utheme) which is in the format specified by the utheme specification, but with a different extension and mime-type. The drawback is the large number of new mime-types that would be created.

Another idea is to specify double extensions: the first relating to the type of theme, and the second utheme, for example: name.metacity.utheme. This would allow easy filtering of files by theme type. I can think of no drawbacks to this, except perhaps the strange file name that would be displayed on Windows systems where extensions are hidden (eg: name.metacity). Applications could choose to create “sub-mime-types” of utheme while retaining the .utheme extension by using this double extension [needs to be confirmed, not possible on Windows at least].

Discussion of program implementation details:

Implementing a thumbnailer for uthemes should be a piece of cake: just extract preview.png from the compressed archive.

Several different implementations for a program that handles uthemes can be created, according to the needs of different users. You could have one for GNOME, one for KDE, one for Xfce, etc. The issue lies in how a program should behave when it encounters a theme type it does not know how to handle. There several ways it could choose to behave:

1. Simply display an error message (eg: “unrecognised theme type: %s”), and possibly ask the user to choose between the options below.
2. Extract the data and open it using the default handler for it by mime-type. (eg: automatically open .emerald themes) The downside to this is that a lot of themes don't have a dedicated mime-type (such as all the ones that are simply gzipped tarballs).
3. Open the utheme in the default archive manager for gzipped tarballs. This is not very user-friendly.
4. Access some sort of central configuration file where programs have registered themselves as utheme handlers for certain theme types, and execute the appropriate command. The advantage of this is that installation, removal and activation commands can be added for new theme types easily, and front-end tools won't have to know about how each theme type is handled.

If number four meets approval we could include the configuration details in another specification.

Last edited by David D. Lowe on 25 Jan 2010. You can contact him at daviddlowe dot flimm at Google's well-known email provider.