GNOME 3 (3.20)
We’ve come a long way

Tobias Mueller
tobiasmue@gnome.org
“GNOME”?
“GNOME 3”?
“GNOME 3.0”?
“GNOME 3.20”?
Elegant design
Activities overview
Notifications
**GtkApplication** is the base class of a Gtk Application. Its primary purpose is to separate your program from `main()`. 

`main()` is an operating system implementation detail that is really uninteresting to applications. The philosophy of GtkApplication is that applications are interested in being told what needs to happen, when it needs to happen, in response to actions from the user. The exact mechanism by which the operating system starts applications is uninteresting.

To this end, GtkApplication exposes a set of signals (or virtual functions) that an application should respond to.

- **startup**: sets up the application when it first starts
- **shutdown**: performs shutdown tasks
- **activate**: shows the default first window of the application (like a new document). This corresponds to the application being launched by the desktop environment.
- **open**: opens files and shows them in a new window. This corresponds to someone trying to open a document (or documents) using the application from the file browser, or similar.

When your application starts, the `startup` signal will be fired. This gives you a chance to perform initialisation tasks that are not directly related to showing a new window. After this, depending on how the application is started, either `activate` or `open` will be called next.

GtkApplication defaults to applications being single-instance. If the user attempts to start a second instance of a single-instance application then GtkApplication will signal the first instance and you will receive additional `activate` or `open` signals. In this case, the second instance will exit immediately, without calling `startup` or `shutdown`.

For this reason, you should do essentially no work at all from `main()`. All startup initialisation should be done in `startup`. This avoids wasting work in the second-instance case where the program just exits immediately.

The application will continue to run for as long as it needs to. This is usually for as long as there are any open windows. You can additionally force the application to stay alive using `g_application_hold()`. In this case, you receive a shutdown signal where you can do any necessary cleanup tasks (such as saving files to disk).
To this end, GtkApplication does not implement main() for you. You must do so yourself. Your main() function should be as small as possible and do almost nothing except creating your GtkWidget and running it. The "real work" should always be done in response to the signals fired by GtkWidget.
Integrated messaging
GtkApplication is the base class for applications in the GTK+ library. It is typically used as the root of an application's implementation, and it provides a number of important functions and signals for managing the application's state.

main() is an operating system implementation detail that is really uninteresting to applications. The philosophy of GtkApplication is that applications are interested in being told what needs to happen, when it needs to happen, in response to actions from the user. The exact mechanism by which the operating system starts applications is uninteresting.

To this end, GtkApplication exposes a set of signals (or virtual functions) that an application should respond to.

- `startup`: sets up the application when it first starts
- `shutdown`: performs shutdown tasks
- `activate`: shows the default first window of the application (like a new document). This corresponds to the application being launched by the desktop environment.
- `open`: opens files and shows them in a new window. This corresponds to someone trying to open a document (or documents) using the application from the file browser, or similar.

When your application starts, the `startup` signal will be fired. This gives you a chance to perform initialisation tasks that are not directly related to showing a new window. After this, depending on how the application is started, either `activate` or `open` will be called next.

GtkApplication defaults to applications being single-instance. If the user attempts to start a second instance of a single-instance application then GtkApplication will signal the first instance and you will receive additional `activate` or `open` signals. In this case, the second instance will exit immediately, without calling `startup` or `shutdown`.

For this reason, you should do essentially no work at all from `main()`. All startup initialisation should be done in `startup`. This avoids wasting work in the second-instance case where the program just exits immediately.

The application will continue to run for as long as it needs to. This is usually for as long as there are any open windows. You can additionally force the application to stay alive using `g_application_halt()`. After you receive a shutdown signal, you can do any necessary cleanup tasks (such as saving files to disk).
Desktop search
with search providers
And more!

- Topic-oriented help
- Intuitive system settings
- ...

- Many small features to improve your experience!
Under the hood

 emploi a famous platform, usable from several languages: C, C++, Python, JavaScript, Vala, ...

 Easy theming with CSS - overhauled
 Symbolic icons
How it all started

Proposal at GUDEAC in Istanbul 2008
Hackfest in Boston
Mockups
TRABAJO CEDIDO AL DOMINIO PÚBLICO

Nosotros Jaime Irving Dávila, Pablo Chamorro, Igor Tamara y Vladimir Tamara hemos escrito "Aprendiendo a aprender Linux: Guía para colegios con plataforma de referencia $PLATFORMA$", que asisten al uso, la instalación y la administración de redes Linux en colegios. Por este medio cedemos esta obra al dominio público, renunciando a todos los derechos patrimoniales con lo que esperamos facilitar su adaptación y uso en colegios.
Butterfly

Superfamily: Hesperiidae

Sippers differ in several important ways from the remaining butterflies, which are classified in the superfamily Hesperiidae and the neotropical superfamily Hedyloidea. Collectively, these three groups of butterflies share several characteristics especially in the egg, larval and pupal stage (Ackery et al., 1999). However, sippers have the antennae clubs hooked backward like a crochet, whilst butterflies have club-like tips to their antennae and hedyloids have feathered or pointy antennae giving them an even more "moth"-like appearance than sippers. Sippers also have generally stockier bodies than the other two groups, with stronger wing muscles. Hesperoidea is very likely the sister group of Hesperiidae, and together with Hedyloidea constitute a natural group or clade.
Actual code
Find a name ;-) 

“Current leader for a project name is the exciting “gnome-shell”

gnomesh

gnomecc
Make it a brand
Make it a brand
Make it a brand
2011–04–06 Release 3.0!
~150 Parties in 48 (sic!) countries

Promotional material produced and shipped
Evolving GNOME 3

http://extensions.gnome.org
GNOME Tweak Tool
From GNOME 3.0 to 3.20

1811 days
339189 changes
GNOME releases

- 2.2: 2003-02-06
- ... 
- 2.26: 2009-03-18
- 2.28: 2009-09-23
- 2.30: 2010-03-31
- 2.32: 2010-09-29
- 3.0: 2011-04-06
- 3.2: 2011-09-28
- 3.4: 2012-03-28
- 3.6: 2012-09-26
- 3.8: 2013-03-27
- 3.10: 2013-09-25
- 3.12: 2014-03-26
- 3.16: 2015-03-25
- 3.18 (G’burg): 2015-09-21
- 3.20 (Delhi): 2016-03-21
The story of contacts (3.2)
The story of contacts (3.4)
The story of contacts (3.8)
The story of notifications
The story of notifications
The story of notifications
The story of notifications
The story of notifications
GNOME 3.20 on March 21th
Approximately 772 people made about 25112 changes to GNOME

- New notification system
- Updated shell visuals
- Improved developer experience
- ...
GNOME is people