开发/Linux下的编译

< Development

过时的翻译将会这样标记。

入门提示

对于新手来说，有一个很常见的问题需要注意：*不要*以超级用户root身份进行编译！那样做会损坏您的编译成果，您需要手动清除所有的内容（包括 ccache 缓存）并且重头开始。以下的示例中，若需要使用超级用户root权限，则会显性地使用 *sudo* 进行区分。

另外，确保您已阅读“一般性的编译提示”。

编译时的依赖项

一般来说，编译 LibreOffice 最简单的方法是在 Linux 下进行。

注意：不支持在 Windows 下的 Linux 子系统中进行编译！

Note:
一般情况下，LibreOffice 官方的源代码分支往往比您的 Linux 发行版预置的版本要新（尤其是 `master` 分支），因此需要更多的编译时依赖项。这种情况下，

> autogen.sh 脚本会告诉您还缺少哪个依赖项。通常，这些额外的依赖项是可选的，并存在对应的 --disable-foo 或者 --without-foo 选项以禁用它；或者，可通过 --without-system-foo 选项使用系统已有的库即可满足该依赖要求（一般该选项为默认）。如果这些方法都无法满足依赖需求，那么您需要在系统中安装一些开发包：先看看 configure 提示的是什么，然后使用 apt-cache search 或者 dnf search 来查找相应的包，这些包通常名称中包含 -dev 或者 -devel 关键字。

Debian 和 Ubuntu

安装依赖项

```
sudo apt-get install git build-essential zip ccache junit4 libkrb5-dev nasm graphviz python3 python3-dev
qtbase5-dev libkf5coreaddons-dev libkf5i18n-dev libkf5config-dev libkf5windowsystem-dev libkf5kio-dev autoconf
libcups2-dev libfontconfig1-dev gperf default-jdk doxygen libxslt1-dev xsltproc libxml2-utils libxrandr-dev
libx11-dev bison flex libgtk-3-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev ant ant-optional
libnss3-dev libavahi-client-dev libxt-dev
```

openSUSE

在 OpenSuse Tumbleweed 下可以使用以下命令安装编译依赖软件包：

```
sudo zypper in git autoconf automake bison make gcc gcc-c++ cups-devel fontconfig-devel gperf java-11-openjdk-
devel libxslt-devel python3-devel krb5-devel libX11-devel libXext-devel libICE-devel libSM-devel libXt-devel
libXrender-devel libXrandr-devel flex gtk3-devel gstreamer-devel gstreamer-plugins-base-devel ant junit nasm
ccache binutils-gold mozilla-nss-devel
```

Fedora/Red Hat

```
sudo dnf builddep libreoffice
```

Arch Linux

以下编译依赖项复制粘贴自官方 Arch 编译系统的 PKGBUILD (https://git.archlinux.org/svntogit/packages.git/tree/repos/extra-x86_64/PKGBUILD?h=packages/libreoffice-fresh)，并进行了一些修改。

```
sudo pacman -S --needed 'base-devel' 'git' 'ccache' 'ant' 'apr' 'beanshell'
'bluez-libs' 'clucene' \
                       'coin-or-mp' 'cppunit' 'curl' 'dbus-glib' \
                       'desktop-file-utils' 'doxygen' 'flex' 'gcc-libs' \
```

```
'gdb' 'glm' 'gobject-introspection' 'gperf' 'gpgme' \
'graphite' 'gst-plugins-base-libs' 'gtk3' \
'harfbuzz-icu' 'hicolor-icon-theme' 'hunspell' \
'hyphen' 'icu' 'java-environment' 'junit' \
'lcms2' 'libabw' 'libatomic_ops' 'libcdr' 'libcmis' \
'libe-book' 'libepoxy' 'libepubgen' 'libetonyek' \
'libexttextcat' 'libfreehand' 'libgl' 'libjpeg' \
'liblangtag' 'libmspub' 'libmwaw' 'libmythes' \
'libnumbertext' 'libodfgen' 'liborcus' 'libpagemaker' \
'libqxp' 'libstaroffice' 'libtommath' 'libvisio' \
'libwpd' 'libwpg' 'libwps' 'libxinerama' 'libxrandr' \
'libxslt' 'libzmf' 'lpsolve' 'mariadb-libs' \
'mdds' 'nasm' 'neon' 'nspr' 'nss' 'pango' \
'plasma-framework' 'poppler' 'postgresql-libs' \
'python' 'qt5-base' 'redland' 'sane' 'serf' 'sh' \
'shared-mime-info' 'ttf-liberation' 'unixodbc' \
'unzip' 'xmlsec' 'zip'
```

如果不准备使用 Java，那么您可以在以上命令中移除 ant, beanshell, java-environment 以及 junit，因为 LibreOffice 大多数情况下没有 Java 也能正常运行。

如果已安装了 'gcc-libs-multilib'，那么您可能需要移除 'gcc-libs' 依赖项。

Slackware

```
./autogen.sh --without-java --disable-postgresql-sdbc --disable-gstreamer-1-0
```

BSDs

- DragonFlyBSD `cd /usr/dports/editors/libreoffice && make patch`
- FreeBSD `cd /usr/ports/editors/libreoffice && make patch`
- NetBSD `cd /usr/pkgsrc/misc/libreoffice && make depends`

克隆并编译

下一步，我们 克隆 源代码仓库，然后开始编译：

```
$ git clone https://gerrit.libreoffice.org/core libreoffice
$ cd libreoffice
```

此处，您可能需要考虑给 autogen 传入更多的编译选项。可用的选项可通过以下命令查看：

```
$ ./autogen.sh --help
```

例如，您可能需要加上调试选项。编译完成后，如果修改了此处的某些选项，那么您可能需要进行完整的重新编译（例如添加了 `--enable-dbgutil` 选项。由于以下的编译命令需要很长时间，因此您可能需要通过添加一些特殊的选项来加速编译，取决于您的最终需求。如果不确定，那么请到 IRC 频道 #libreoffice-dev 询问。

You should have a suitable configuration for the build that you put inside `autogen.input` file. But, you can start with a minimum set of options and customize them later. A sample `autogen.input` for building LibreOffice on Linux can be as follows, to enable debug build, and also disable doxygen to save time. Since the `make` command can take a vast quantity of time to run for the first time, these options are helpful.

```
--enable-dbgutil
--without-doxygen
```

Then you can prepare build files using:

```
$ ./autogen.sh
```

Now you are ready for building using `make`. Beware that full (all components, non-incremental) build may take up to 24 hours and takes up about 20 GB of drive space. You will also most likely need at least 8 GBs of RAM, otherwise the machine might fall into swap and appear to freeze up.

It is a good idea to first run `make`, so you have a working build in case `make check` later fails.

```
$ make
$ make check
```

运行编译好的程序

编译完成后，您将得到可以正常运行的二进制文件，位于 instdir 中，您可以通过以下命令来运行：

```
$ instdir/program/soffice --writer
$ instdir/program/soffice --calc
```

视频教程

首次编译视频教程（墙外）

Please accept this video. By accepting you will be accessing content from YouTube, a service provided by an external third party.

YouTube privacy policy

Accept YouTube Content

尽管该介绍是在Ubuntu系统中进行的，但是它经过很少的修改就能在任何其它Linux系统上工作（若有需要请见下方的说明）。我们建议您一边跟随这些步骤，一边观看视频。

提示

在调试器下运行

直接在调试器中启动您编译好的 LibreOffice：

```
$ make debugrun
```

在调试器中启动后，您可以通过以下命令运行 Writer 组件：

```
   (gdb) run --writer
```

需要注意的是，首次通过这种方式（或通过 make debugrun）启动时，*soffice.bin* 进程会终止，这是正常现象。您只需要再次运行即可。

启用“警告”

多数版本的 GCC 编译器在高级别的优化编译选项时经常会忽略一些它认为无用的“警告”信息。不要将 `--disable-debug --enable-werror` 选项在一起使用。

性能

编译 LibreOffice 很耗时，非常耗时。具体需要多长时间，取决于您的机器性能有多强大。但是，不管怎样，我们有一些工具可以从不同层面来加速编译过程。

使用 ccache 缓存以加快重新编译的速度

ccache (http://ccache.samba.org/) 是一个工具，它可以将 C/C++ 的编译结果缓存起来，以备后续复用。对于首次编译，该工具没什么用（实际上它会稍微托盘首次编译过程），但是它能显著加快重新编译的速度。如果您准备更改 LibreOffice 源代码中的许多头文件，那么这个工具值得拥有。

默认情况下，当 autogen.sh 脚本检测到您的系统中已经安装了 ccache 时，会自动启用它。要获得最好的效果，您可能需要将其缓存大小限制调大，或者启用缓存压缩。您可以通过 `man ccache` 来查看 ccache 的具体配置选项。

ccache 的默认最大缓存限制为 5GB，这对于 LibreOffice 的编译来说太小了。对于不包含调试符号的编译，至少应当有 8GB 缓存空间；对于有调试符号以及其他调试选项的编译，也许 32GB 会比较合理。您可以通过以下命令来调整缓存大小（请注意，该选项会保存到缓存目录，因此如果您后续更改了 `$CCACHE_DIR` 缓存目录位置，则需要重新设置缓存大小）：

```
ccache --max-size 32G
```

ccache 支持压缩。如果您的磁盘空间比较小，或者使用的是 SSD 固态硬盘或者笔记本硬盘，那么启用压缩是个不错的选项。您可以通过以下命令来启用 ccache 压缩：

```
export CCACHE_COMPRESS=1
```

使用 Icecream/distcc 进行分布式编译

如果您局域网内有多台机器，那么您可以使用 icecream (http://people.gnome.org/~michael/blog/icecream.html) 或者 distcc (http://distcc.samba.org/) 工具进行分布式编译。我们推荐使用 Icecream，它更容易配置。

LibreOffice 内置了对 Icecream 的支持，您只需要在运行 `./autogen.sh` 脚本时加上 `--enable-icecream` 选项（或者将其加入到源代码根目录下的 autogen.input 文件中），编译时 configure 进程会自动配置可用的任务数量，并在 `/usr/lib/icecc/bin` 或者 `/opt/icecream/bin` 下寻找 icecream 的 gcc 包装器。如果您将其安装到了其它位置，那么需要通过 `--with-gcc-home=/path/to/your/icecream/bin` 来指定其路径。

之后，在主编译机器上配置 `distcc` 从而能够使用其它的机器进行编译。然后，为 autogen.sh 加上 `--with-parallelism=N` 选项，此处 N 表示您的编译集群中的 CPU 数量。最后，运行 `distcc-pump make CC=distcc CXX=distcc` 来启动分布式编译。

预编译的头文件

为 autogen.sh 传入 `--enable-pch` 可以在编译时使用预编译的头文件 (precompiled headers, PCH), 这将减少编译耗时，但同时会占用更多磁盘空间、并且在头文件有变更时会有更多的源代码被重新编译。该选项有几个级别：system, base, normal, full. 更高的级别意味着更少的编译耗时、更多的磁盘空间占用、以及源代码更改时更多的重新编译可能性。

--with-parallelism（并行编译）

编译过程中可以指定并行编译的任务数量。任务数量由 autogen.sh 的 --with-parallelism 选项进行控制。 如果内存空间足够，我发现使用 --with-parallelism=n（此处 n 为您的 CPU 核心数量），或者 --with-parallelism=2（如果您的 CPU 只有一个核心），可以获得最快的编译时间。

--with-parallelism 默认即等于您的系统上的 CPU 核心数。如果您使用 --enable-icecream，则默认为 10.

--with-system-libs（使用系统库文件）

编译时使用系统库文件可以加快编译速度，但是您需要搞定依赖项问题。

编译耗时估计

在现代的机器上，首次编译或者一次“干净”的重新编译大约需要 8/核心数 小时。 若使用了 ccache, 后续的每日编译耗时大约需要 2 小时到 10 分钟，取决于源代码更改的程度及其复杂性。

让 make 显示详细日志

Make 默认会隐藏其具体的命令调用细节。如果想知道 make 究竟干了些什么，您可以通过以下选项启用它：

```
$ make verbose=1
```

Using `verbose=1`, in turn sets this option for the child `make` invocations, and also deals with externals. In the verbose mode, every single command used to build LibreOffice is shown, which can be used for troubleshooting build problems.

在多个工作目录间切换

如果您想在 master 分支以及当前的发布版分支上同时工作，那么您可以共享 git 仓库以及所用的外部源代码包 (external source tarballs)。这会显著节省时间以及磁盘空间占用，尤其是当启用庞大的 l10n 仓库的情况下。请注意，在同一个编译目录下，我们不建议在不同的分支之间通过 git 检出的方式来回切换，这是因为即便没有依赖项问题，其编译耗时也会跟一次“干净”的重新编译差不多。如果您能承受更大的磁盘空间占用，那么推荐的方式是对不同的分支使用单独的目录树。

有两种方式可以在不同的分支间共享 git 仓库：1. `git clone --reference`（引用式克隆）；2. `git-new-workdir`. 对于第一种方式，“引用式克隆”，它不会重新去下载整个 git 仓库，同时新的工作目录中会有其单独的 .git/config 等配置文件。这种方式受 git 上游开发者支持。对于第二种方式，git-new-workdir, 它会当您在两个不同的工作目录中检出同一个分支时（或者其他的操作中）导致很多问题，因此不被支持。

设置

要配置“引用式克隆”，您首先需要按照上述方法进行首次编译。对于第二次以及其他编译，可通过以下方式来克隆原始的 core 核心仓库：

```
$ git clone --reference /path/to/master --branch name-of-branch
ssh://logerrit/core /dir/to/be/created
```

此时，您会注意到克隆速度非常快。之后，在克隆得到的新的工作目录中进行如下额外的配置（或者叫 autogen.sh 选项）：

```
$ ./autogen.sh ... --with-referenced-git=/path/to/master --with-external-
tar=/path/to/master/external/tarballs
```

此时克隆过程将会很快。您可能需要为 --with-external-tar 选项使用绝对路径。在这之后，您可以像在单独的工作目录中进行编译一样进行编译步骤。

此处的 --with-referenced-git 选项尽在您使用了一些子模块（比如 translations 翻译、 dictionaries 词典、或者 help 帮助）、或者是编译4.0分支之前的老版本时才有用。在新的版本中，子模块默认被禁用。

如果需要在不同的工作目录之间进行 cherry-pick 操作，那么您可以设置您的 git 本地副本，让其像 remote 一样工作，就像这样：

```
$ cd /path/to/stable-workingcopy
$ git remote add unstable /path/to/master/.git
$ cd /path/to/master
$ git remote add stable /path/to/stable-workingcopy/.git
```

然后您就能像使用真正的远程 remote 服务器进行 cherry-pick 一样进行操作了。

检出一个发布版分支

在 master 分支上以需要的选项运行 autogen.sh，然后运行 make fetch 以抓取所需的子模块（取决于您传给 autogen.sh 的选项）。

然后通过以下命令切换分支：

```
$ ./g checkout -b libreoffice-4-0 origin/libreoffice-4-0
```

rpm 未找到 / ant 未找到 / 不存在 gnome-vfs-2.0 包

--disable-epm 将解决 "rpm not found" 错误。这是因为，在启用 epm 的情况下 autogen.sh 会在您的系统上查找 dpkg 或者 rpm。

--without-java 将解决 "ant not found" 错误。

在编译 UNO SDK (http://api.libreoffice.org) 时，您需要安装 Doxygen (http://www.doxygen.org)，从而能够生成 C++ UNO 接口的 HTML 格式的文档。一个选择是，从 官方网站 (http://www.doxygen.org) 安装 Doxygen，并确保 doxygen 可执行文件能在 PATH 路径中找到到，或者手动通过 --with-doxygen=路径 编译选项指定具体路径。另一个选择是采用 --without-doxygen 编译选项，此时在编译 UNO SDK 时将不会生成 C++ UNO 接口的 HTML 文档。第三个选择是采用 --disable-odk 编译选项，此时将不会编译 UNO SDK。

备注： Ubuntu 下安装 'libgnomevfs2-dev' 包将解决 "No package 'gnome-vfs-2.0' found" 错误。该问题仅在老版本中存在。

```
$ sudo apt-get install libgnomevfs2-dev
```

构建 DEB 和 RPM 包

如果您需要生成RPM或DEB包，则需要在您的的autogen.input文件中加入以下选项：

RPM 和 DEB

RPM

```
--with-package-format=rpm
--enable-epm
```

DEB

```
--with-package-format=deb
--enable-epm
```

文档基金会的发行版基本编译配置位于：distro-configs 模块 (https://docs.libreoffice.org/distro-configs.html)。对于Linux，请参考：distro-configs/LibreOfficeLinux.conf (https://git.libreoffice.org/core/+/refs/heads/master/distro-configs/LibreOfficeLinux.conf)。

编译好的二进制包在哪儿?

位于 ./workdir/installation/ 目录下。

处理行尾结束符以及 git 的 autocrlf

如果 autogen.sh 输出类似下面的提示：

```
./autogen.sh: line 1: $':\r': command not found
' '--srcdir=/home/user/libreoffice' '--enable-option-checking=fatal'
configure: WARNING: you should use --build, --host, --target
configure: WARNING: invalid host type:
************************************************************************
*
*   Running LibreOffice build configuration.
*
************************************************************************

' not recognized system type... Invalid configuration `
 failedre: error: /bin/bash ./config.sub
Error running configure at ./autogen.sh line 241.
```

... 这种情况下，您很可能遇到了行尾结束符问题。发生这种现象，很可能是您将错误地设置了 git 的 core.autocrlf=true.

运行以下命令来解决该问题： (**警告：未提交的变更将会丢失！！**):

```
git config --unset core.autocrlf
git rm --cached -r .
git reset --hard
git clean -x -f
./autogen.sh
```

使用 Weblate 上的 po 文件编译已翻译的用户界面

LibreOffice 的用户界面是使用 .po 文件在 Weblate 服务器 (https://translations.documentfoundation.org/) 上翻译的。翻译的成果在每一次新的版本发布时会以语言包的形式包含。这往往意味着已发布的版本中的翻译没有体现各版本之间的翻译改进。另外，每日构建版 (https://www.libreoffice.org/download/pre-releases/) 仅提供对为数不多的几个语言版本。作为翻译者，您可能需要在各个版本发布前测试翻译效果，因此在需要编译时从 Weblate 上抓取 PO 文件。

首先，确保您已经解决了编译时的依赖软件包问题，然后克隆并开始编译。

编译已翻译的用户界面

1. 首先，从 Weblate 上下载 .po 文件（需要登录才能下载）。
2. 确保在启用语言包的情况下已经成功进行了一次编译（即使用了类似 ./autogen.sh --with-lang="zh-CN" 进行过了编译）。
3. 解压缩已下载的 .po 文件，复制到源代码的相应目录中：cp -R zh-CN/libo_ui/* libreoffice/translations/source/zh-CN/

4. 接下来是最棘手的一步：获取到的 .po 文件，有一些名称是 'messages.po'，而另外一些是其他的名称。您需要做的是，将所有的 .po 文件转换为 .mo 文件，但需要注意：

    1. 对于名称 **不是** 'messages.po' 的文件，通过运行 `make build-l10n-only` 自动将其转换为 .mo 文件。
    2. ## 对于名称 **是** 'messages.po' 的文件，您需要通过以下命令手动将其转换为 .mo 文件：`msgfmt -o instdir/program/resource/zh-CN/LC_MESSAGES/<moFile> <dir>/messages.po`，对于每一个这样的文件都需要进行这样的转换。注意，此处的语言代码使用的是zh-CN，若需要其他的语言代码则需要替换；此处的 <moFile> 对应下表中的左侧一列的内容，<dir> 对应右侧一列的内容；请确保 '<dir>/messages.po' 的路径是从当前工作目录到最终目标文件的完整路径。

如以上第4步所说，一些 .po 文件被命名为 'messages.po' 而另一些是别的名称。下表中列出了可能包含了 'messages.po' 文件的目录，对于这些文件需要采用第 4.2 步将其转换为 .mo 文件：

<mofile>	<dir>
acc.mo	accessibility
avmedia.mo	avmedia
basctl.mo	basctl
chart.mo	chart2
cnr.mo	connectivity
cui.mo	cui
dba.mo	dbaccess
dkt.mo	desktop
editeng.mo	editeng
filter	filter
for.mo	formula
fps.mo	fpciker
frm.mo	forms
fwk.mo	framework
pcr.mo	extensions
rpt.mo	reportdesign
sb.mo	basic
sc.mo	sc
sca.mo	scaddins
scc.mo	sccomp
sd.mo	sd
sfx.mo	sfx2
sm.mo	starmath
svl.mo	svl
svt.mo	svtools
svx.mo	svx
sw.mo	sw
uui.mo	uui
vcl.mo	vcl
wiz.mo	wizards
wpt.mo	writerperfect
xsc.mo	xmlsecurity

最后一步是运行 `sudo make install` 以安装包含了新增翻译的版本，或者通过 `./instdir/program/soffice` 来运行程序以检查新翻译在界面上的情况。

如果您需要编译 master 分支然后检查界面翻译效果, 则可能需要运行 "make translations" 以将 .po 文件与 .pot 翻译模板文件进行同步合并。

编译已翻译的帮助文档

帮助文档也是在 Weblate 上翻译的，因此对于已翻译的帮助文档，可以执行以下操作：

1. 下载已翻译的帮助文档的 .po 文件。
2. 运行 `./autogen.sh --with-help --with-lang="<语言代码>"`。注意将 <语言代码> 替换为实际的语言，比如 zh-CN。如果需要将帮助文档编译为 HTML 文件，请使用 `--with-help=HTML` 选项。
3. 运行 `make`
4. 运行 `cp -fR <语言代码>/libo_help/* libreoffice/translations/source/<语言代码>/helpcontent2/source/text/`，注意将两处 <语言代码> 替换为实际的语言代码，并确保该路径指向了从当前工作目录能够到达的正确的、确实存在的目的文件夹。
5. 运行 `mv libreoffice/translations/source/<语言代码>/helpcontent2/source/text/auxiliary.po libreoffice/translations/source/<语言代码>/helpcontent2/source/`. 注意将两处 <语言代码> 替换为实际的语言，并确保该路径指向了从当前工作目录能够到达的正确的、确实存在的目的文件夹。
6. 运行 `make build-l10n-only`
7. 运行已编译好的 LibreOffice.

(感谢 Kiyotaka 和 Jihui)
