Using GNU Guix for software development

Dr. Arne Babenhauserheide

<2020-02-19 Mi>

When I started working as a software developer at Disy, I also switched my main distribution at home to GNU Guix.

Guix enables me to run updates without fearing that losing power during the update could kill my system (I had that happen with Gentoo), it provides a stable base-system with per-user installed packages as well as environments that can install packages into the local shell much like language-specific package managers like npm. Also it aligns well with my ideals of using purely free software.

But I mostly do proprietary software development at work, and advanced features of Guix make some of that problematic. This page lists the workarounds I'm using.

Contents

IntelliJ (Ultimate)

We're using IntelliJ for Java development, because it is the best tool out there for a 1.5 Million lines Java codebase. But it requires running software which is not installed via the package manager.

I run it via the file `~/.local/bin/idea.sh` which sets a custom LD_LIBRARY_PATH and the system-JDK, and then runs the idea.sh script which comes with IntelliJ.

Part of the library path is required for our starters, because of native code that has to find the libraries and libstdc++.

Update: Cleaned up finding the path to libstdc++ with Guix onboard tools.

```
#!/bin/bash

cd ~/

# need openjdk 14, because 16 stops with module errors
JAVA_PACKAGE="openjdk@14"
GCC_TOOLCHAIN="gcc-toolchain@10.3"
JDK_PATH="$(guix build ${JAVA_PACKAGE} | grep -- '-jdk$')"
# need to track libstdc++ in the dependencies of the GCC toolchain
GCC_LIB_PATH="$(grep -oE "[^\"]*gcc-10[^\"]*-lib" $(grep -oE "[^\"]*gcc-10[^\"]*dr
SQLITE_PATH="$(guix build sqlite | head -n 1)"
SPATIALITE_PATH="$(guix build libspatialite)"
exec -a "$0" guix environment --ad-hoc ${JAVA_PACKAGE} ${JAVA_PACKAGE}:jdk ${GCC_T
```

For info on the (no longer necessary) trap, see http://redsymbol.net/articles/bash-exit-traps/

The fork-hack to run code after exec is from that other guy on stackoverflow.

Maven

I run a custom version of Maven via a simple script:

```
#!/bin/sh

NOTE='\033[1;33m'
NONE='\033[0m'

echo "[${NOTE}NOTE${NONE}] Running Maven with:"
java -version

~/Disy/opt/apache-maven-3.6.1/bin/mvn "$@"
```

npm

npm is a bit of a beast: If I use what we create via maven, it dies because it does not find its libraries. Therefore I install it manually with the system-installed npm and then run `npm-cli.js` directly:

```
npm install npm@6.13.4 ; node_modules/npm/bin/npm-cli.js install; node_modules/npm
```

To simplify this, I moved it into a shell-script at `~/.local/bin/npm`:

```
#!/usr/bin/env bash
# if ! echo $PWD | grep -q Disy; then
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/home/arne/.guix-profile/lib:/gnu/store/
exec -a "$0" guix environment --ad-hoc gcc-toolchain@10.2 -- /home/arne/.guix-prof
```

Firefox

Update 2020-10: There are currently no updates to the flatpak package. I get the warning "Info: org.gnome.Platform is end-of-life, with reason: The GNOME 3.34 runtime is no longer supported as of 14th August 2020. Please ask your application developer to migrate to a supported platform."

I actually tried to package Firefox for Guix, but packaging a few hundred Rust packages is quite some task. I stopped at more than 200 packaged ones, because the version numbers retrieved from `guix import crate NAME` did not match the version numbers needed but rather always retrieved the most recent version.

Though it ultimately failed, this is the script I used to automatically import many Rust packages:

```
for i in $(for i in $(grep -o ',[^ ()][^ ()]*' more/packages/gnuzilla.scm | sed s.
       grep -q "name \"$i\"" more/packages/gnuzilla.scm || echo $i;
   done | sort -u | grep rust | sed s/^rust-//  | xargs); do
    guix import crate $i >> more/packages/gnuzilla.scm || \
(sleep 30 && guix import crate $i >> more/packages/gnuzilla.scm);
done
```

The actual solution to get Firefox was to install flatpak and get Firefox from there:

```
guix install flatpak
flatpak remote-add --user --from org.mozilla.FirefoxRepo https://firefox-flatpak.m
flatpak install --user org.mozilla.FirefoxRepo org.mozilla.FirefoxDevEdition
```

Now I run Firefox via a small script at `~/.local/bin/firefox`:

```
#!/run/current-system/profile/bin/bash
flatpak run org.mozilla.FirefoxDevEdition "$@"
```

Zoom

For Zoom I use the same method as for Firefox:

```
flatpak install flathub us.zoom.Zoom
```

```
#!/run/current-system/profile/bin/bash
flatpak run us.zoom.Zoom
```

Custom Emacs profile

All my planning files and my setup to develop Javascript are in a specialized Emacs setup that I run with `disymacs`:

```
#!/usr/bin/env bash
HOME="${HOME}/Disy" emacs "$@"
```

There is a full custom setup in `~/Disy/.emacs.d` so I can more easily synchronize the config between office and home office.

Connect to VPN

I use `openconnect` and a custom script to connect to our VPN.

```
guix install openconnect
```

```
#!/run/current-system/profile/bin/bash
if [ "$EUID" -ne 0 ]; then
  echo This script needs root priviledges. 1>&2
  echo Executing sudo --login $(realpath "$0") in 3 seconds 1>&2
  for i in {1..3}; do
      echo -n .
      sleep 1
  done
  echo " " now executing sudo --login $(realpath "$0")
  exec sudo --login $(realpath "$0")
fi
# Connect via VPN to disy and ensure that the nameservers are correct
```

```
function resolv-disy-undo {
    echo "undoing VPN connection"
    cp /etc/resolv.conf.head-default /etc/resolv.conf.head
    # ensure that the manual DNS servers are available
    # let dhcp do the rest # TODO: this only worked in Gentoo. Find out how it wor
    cp /etc/resolv.conf /tmp/resolv.conf && cat /etc/resolv.conf.head /tmp/resolv.
    # /lib/dhcpcd/dhcpcd-run-hooks
}

# ensure that the change is undone when this script ends, see http://redsymbol.net
trap resolv-disy-undo EXIT HUP INT QUIT ABRT TERM KILL

if test ! -e /etc/resolv.conf.head-default; then
    cp /etc/resolv.conf.head /etc/resolv.conf.head-default || exit 1
fi

# /etc/resolv.conf.head-disy contains our internal name servers,
# one of them duplicated, because dhcpcd only uses the first three nameservers.

cp /etc/resolv.conf.head-disy /etc/resolv.conf.head
# /lib/dhcpcd/dhcpcd-run-hooks # update /etc/resolv.conf

echo
echo When requested, type your PASSWORD directly followed by the AUTHENTICATOR-TOK
echo

(sleep 30 && sshfs -o allow_other,defer_permissions USER@NODE:/path/to/disk /path/

openconnect --protocol=gp gp.disy.net --csd-wrapper="$HOME/.guix-profile/libexec/c

resolv-disy-undo
reset
```

Run tomcat with Java 8

I run my system with OpenJDK 12, and we develop with Java 11, but our secondary Tomcat setup needs Java 8. So I create a local environment with Java 8 (via icedtea 3.7).

```
# download and unpack tomcat and enter the folder, then call
guix environment --ad-hoc icedtea@3.7.0
bin/startup.sh
```

Versiontracking via magit and monky

To make git usable I use magit. It gives me a good balance of efficiency and control.

For Mercurial, my preferred version tracking tool, I now use monky to have similar integration into Emacs as with magit.

Full update of installed packages

`guix pull && guix update -k` can be slow if substitutes cannot be found, so everything has to be built locally. I still don't know why that happens, but I now have a way to avoid it: I just re-install all installed packages.

```
#!/usr/bin/env bash
# re-install all guix packages
guix pull && guix package -I | cut -f 1,3 | sed 's/\t/:/' | xargs guix install
```

I saved this as `~/.local/bin/update-guix-full`.

Alfaview

For lectures with so many people that jitsi and nextcloud meet their limits. European alternative to Zoom.

First get the debian image and unpack it. Then unpack the data tarball.

```
#!/usr/bin/env bash
cd ~/Downloads/alfaview/data/opt/alfaview || exit 1
exec -a "$0" guix environment --ad-hoc gcc-toolchain@10.2 -- bash -c "LD_LIBRARY_P
```

Adapt the paths, then createthis file as `~/.local/bin/alfaview` and make it executable with `chmod +x ~/.local/bin/alfaview`.

Unlock cores

In my Guix config I throttle my cores to reduce power consumption. But for work I often need full performance in short bursts — for example when IntelliJ re-indexes sources or when I rebuild after changes. For this, I have a small core-unlocker script saved at `~/.local/bin/boost-cpu`:

```
#!/usr/bin/env bash
# boost CPU for MIN minutes
if [[ x"$1" == x"" ]]; then
    echo "Error: missing argument."
    echo "usage: $0 <minutes>"
    exit 1
fi
if ! test $1 -eq $1 2>/dev/null; then
    echo "Error: argument '$1' is not a number."
    echo "usage: $0 <minutes>"
    exit 1
fi
if [[ x"$1" == x"--help" ]]; then
    echo "$0 <MIN>: boost CPU speed for MIN minutes."
    exit 0
fi

SEC="$(($1 * 60))"

for i in $(seq 1 $SEC); do sudo cpupower frequency-set -u 6000000; sleep 1; done
```

Test against IE11 in a virtual machine

First get one of the IE11 VMware images from Microsoft: https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/

```
wget https://az792536.vo.msecnd.net/vms/VMBuild_20190311/VirtualBox/MSEdge/MSEdge.
```

Unzip and untar it:

```
unzip MSEdge.Win10.VirtualBox.zip
tar xvf MSEdge\ -\ Win10.ova
```

Then convert if for use with qemu:

```
qemu-img convert -f vmdk IE11-Win81-VMWare-disk1.vmdk -O qcow2 W81.qcow2 -p
```

Finally run the VM and access your local service over your internal (NAT-) IP (use `ifconfig` to get that):

```
qemu-system-x86_64 -accel kvm -smp 4 -m 8000 -hda W81.qcow2
```

If you organize with org-mode, you can create a link that runs Windows when you open it (with `C-c C-o` or by clicking it) like this:

```
[[shell:cd /mnt/blattwerk/downloads && qemu-system-x86_64 -accel kvm -smp 4 -m 800
```

Get the (OWA) Exchange-Calendar in the org-agenda via ICS

In OWA open the **Options**. Then select under Calendar and sharing the **publish calendar** option. Choose the calendar and **export all details**. Then copy the link to the **ICS file**.

For this example let's assume that it is http://ccc-ffm.de/wp-content/uploads/2015/02/Vortrag-FSFE-Freie_Software_in_der_Bildung.ics.

Now add the following to your `~/.emacs.d/init.el`:

```
;; Add calendar, option 3  https://www.ict4g.net/adolfo/notes/emacs/emacs-caldav.h
(unless (file-exists-p "~/.emacs.d/diaries/")
  (make-directory "~/.emacs.d/diaries/"))
(setq diary-location "~/.emacs.d/diaries/")
(add-hook 'diary-list-entries-hook 'diary-include-other-diary-files)
(add-hook 'diary-mark-entries-hook 'diary-mark-included-diary-files)
(setq calendars
      '(("exchangeexport" . "http://ccc-ffm.de/wp-content/uploads/2015/02/Vortrag-
(defun getcal (url file)
```

```
  "Download ics file and add it to file"
  (let ((tmpfile (url-file-local-copy url)))
    (icalendar-import-file tmpfile file)
    (kill-buffer (car (last (split-string tmpfile "/"))))))
(defun getcals ()
  "Load a set of ICS calendars into Emacs diary files"
  (interactive)
  (save-mark-and-excursion
    (save-window-excursion
      (with-silent-modifications
(mapcar #'(lambda (x)
  (let ((file (concat diary-location (car x)))
(url (cdr x)))
    (message (concat "Loading " url " into " file))
    (find-file file)
    ;; (flush-lines "^[& ]") ;; if you import ical as non marking
    (erase-buffer) ;; to avoid duplicating events
    (getcal url file)))
calendars)))))
;; set your export as diary file (I did not get the import from the general diary
(setq org-agenda-diary-file "~/.emacs.d/diaries/exchangeexport")
;; optional: add getcals to midnight-mode so it updates every night
(add-to-list 'midnight-hook 'getcals)
```

Finally include the new calendar in your diary-file, so you see the entries in `M-x diary` and the `M-x calendar`:

```
echo '#include "diaries/exchangeexport' >> ~/.emacs.d/diary
```

For desktop notifications, look into `org-agenda-to-appt`.

Fix font support in Chromium

```
xset +fp $(dirname $(readlink -f ~/.guix-profile/share/fonts/truetype/fonts.dir))
fc-cache -rv
```

Eslint

Save the following as `~/.local/bin/eslint`:

```
#!/usr/bin/env bash
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/home/arne/.guix-profile/lib:/gnu/store/
exec -a "$0" guix environment --ad-hoc gcc-toolchain@10.3 -- /home/arne/.guix-prof
```

Eslint requires libc in the path. Since I did not find how to get only the gcc library as package, I search for the gcc package used by the gcc-toolchain via

```
ls -d /gnu/store/*gcc-10.3.0-lib/lib/ &
```

Note the version `10.3`.

run npm node binary

```
GCC_TOOLCHAIN="gcc-toolchain@10.3"
GCC_LIB_PATH="$(grep -oE "[^\"]*gcc-10[^\"]*-lib" $(grep -oE "[^\"]*gcc-10[^\"]*dr
guix environment --ad-hoc nss node ${GCC_TOOLCHAIN} sqlite libspatialite node -- b
  "LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${GCC_LIB_PATH}/lib:$(realpath ~/.guix-profi
```