Building, installing and running software

Day one

Bob Dowling
University Computing Service

http://www-uxsup.csx.cam.ac.uk/courses/
http://training.csx.cam.ac.uk/

Why do this course?

It's my supervisor's code.

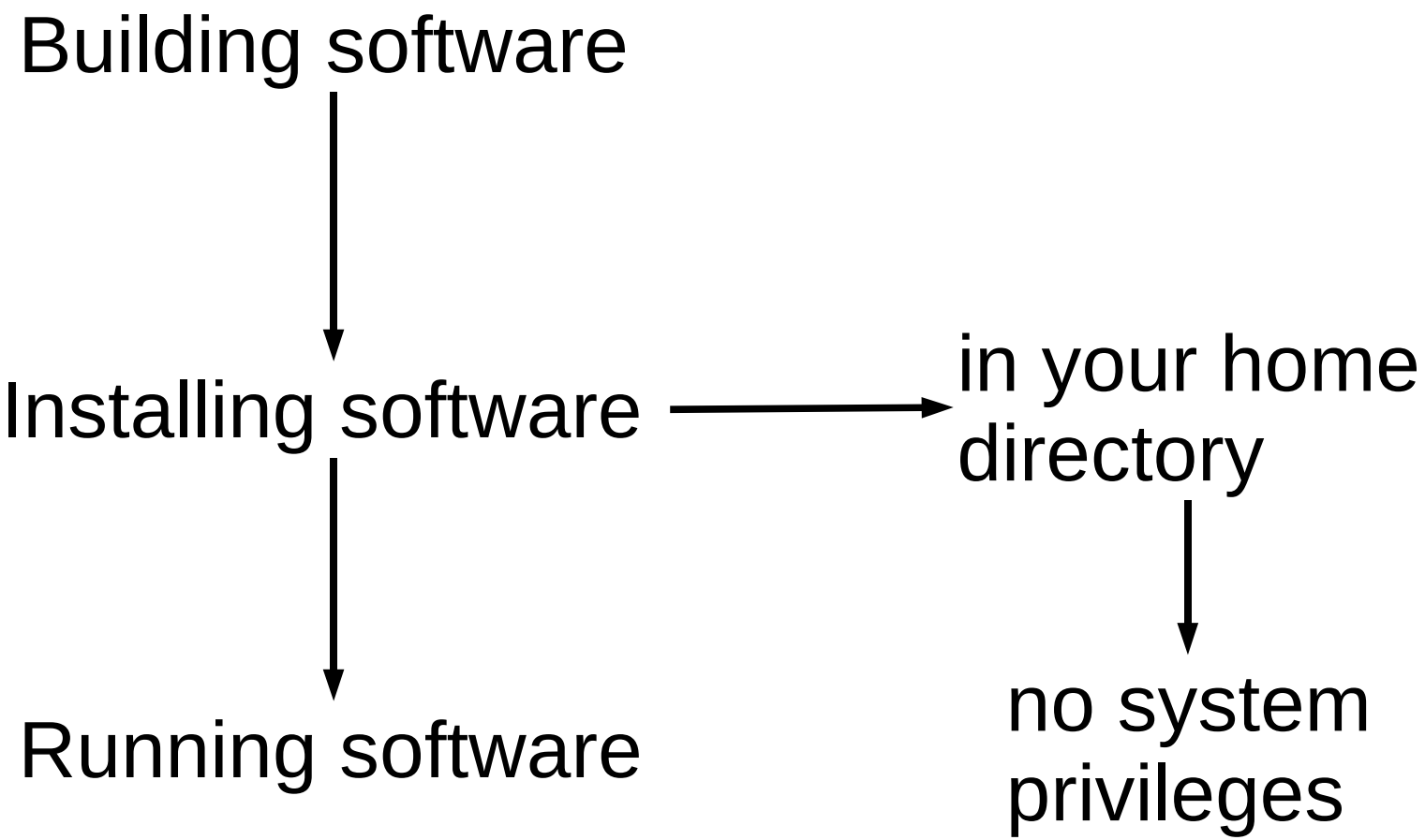
It's the standard software in the … community.

“**other people's software**”

I need a newer version.

I found it on Google.

What will you learn?



What is this course *not* for?

System administration

System directories

Writing software

Course outline

Location Unpacking “Configured” builds	1st afternoon
make and Makefiles Software libraries	2nd afternoon
“Real world” example Recursive make	3rd afternoon

Location?

```
${HOME}/bin
```

programs

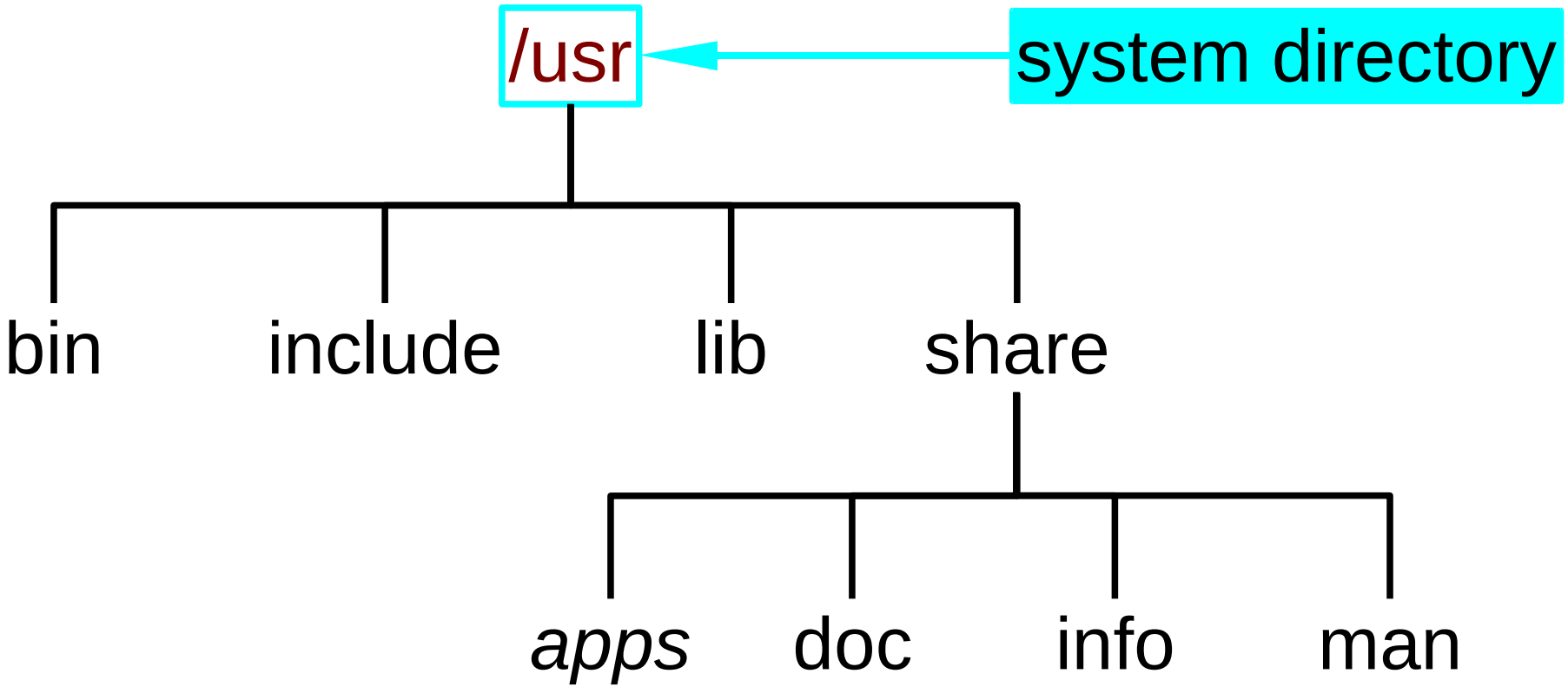
But what about…

documentation?

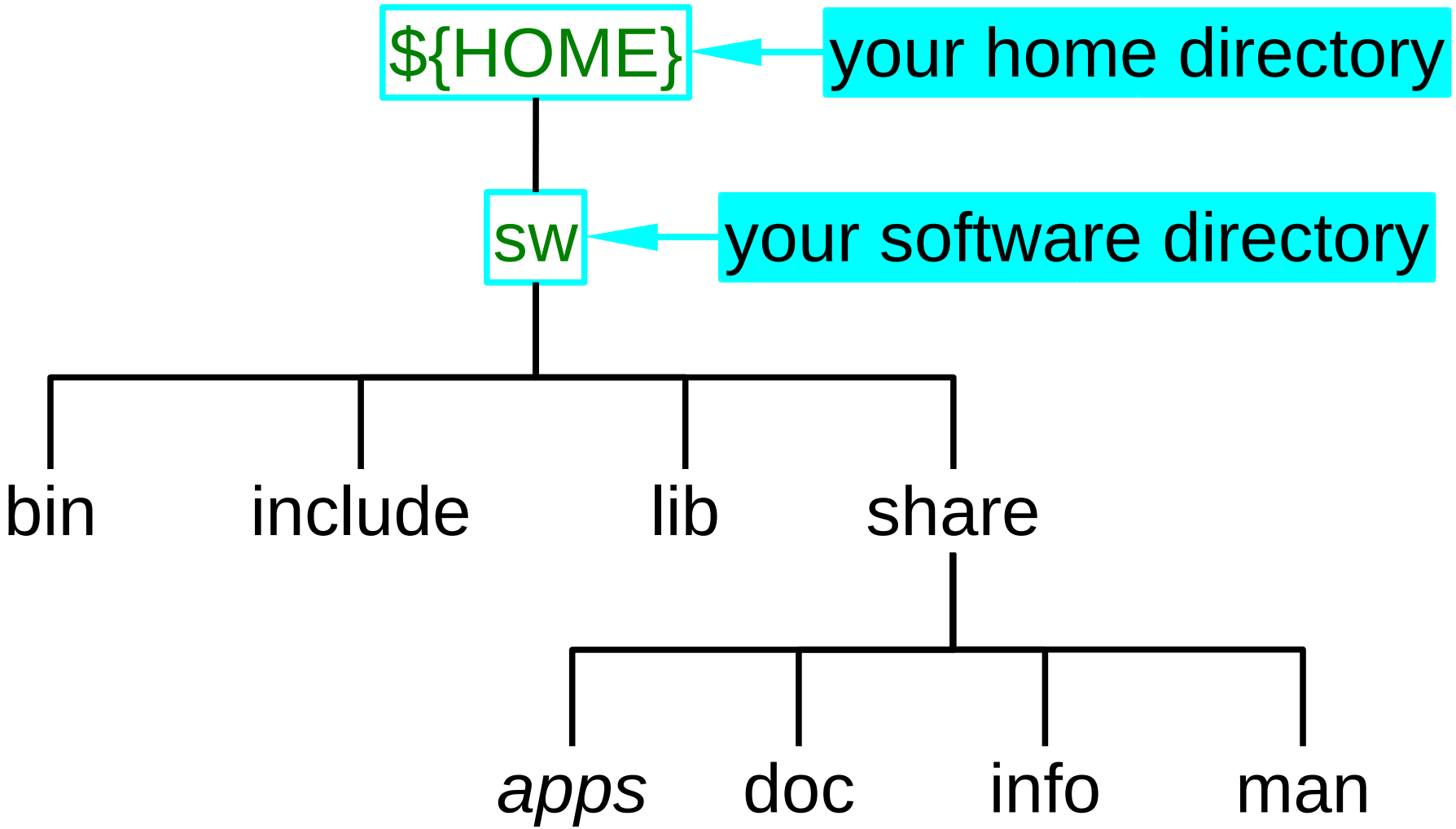
graphics?

libraries?

Mimic the system location



Mimic the system location



Exercise

Some software needs the tree to exist before it can be installed.

We will build the tree ourselves.

```
$ /ux/Lessons/Building/mkswtree
```

Finding programs

Environment variable: PATH

```
$ echo ${PATH}
/usr/local/bin:/usr/bin:/bin:
/usr/bin/X11:/usr/X11R6/bin:
/usr/games:/opt/kde3/bin:
/usr/lib/mit/bin:/usr/lib/mit/sbin:
/opt/novell/iprint/bin:
/opt/real/RealPlayer
```

```
/usr/local/bin:/usr/bin:/bin:
/usr/bin/X11:/usr/X11R6/bin:
/usr/games:/opt/kde3/bin:
/usr/lib/mit/bin:/usr/lib/mit/sbin:
/opt/novell/iprint/bin:
/opt/real/RealPlayer
```

/usr/local/bin/ls ✗

/usr/bin/ls ✗

/bin/ls ✓

Modifying PATH

“paranoia” quotes

```
export PATH="${PATH}/sw/bin:${PATH}"
```

alter an environment variable

this env. var!

our new directory

the old PATH value

Setting PATH automatically

`${HOME}/.bashrc`

File automatically run every time you log in.

We will put the command there.

NB: **Only** when you start a session.

Not just PATH !

commands	PATH
`$ ls`	`${HOME}/sw/bin`
manual pages	MANPATH
`$ man ls`	`${HOME}/sw/share/man`
information pages	INFOPATH
`$ info ls`	`${HOME}/sw/share/info`

Exercise

1. Copy in a new ${HOME}/.bashrc file.

```
/ux/Lessons/Building/bashrc1
                         └──→${HOME}/.bashrc
```

2. Copy in a new command.

```
/ux/Lessons/Building/hello
                         └──→${HOME}/sw/bin/hello
```

Exercise

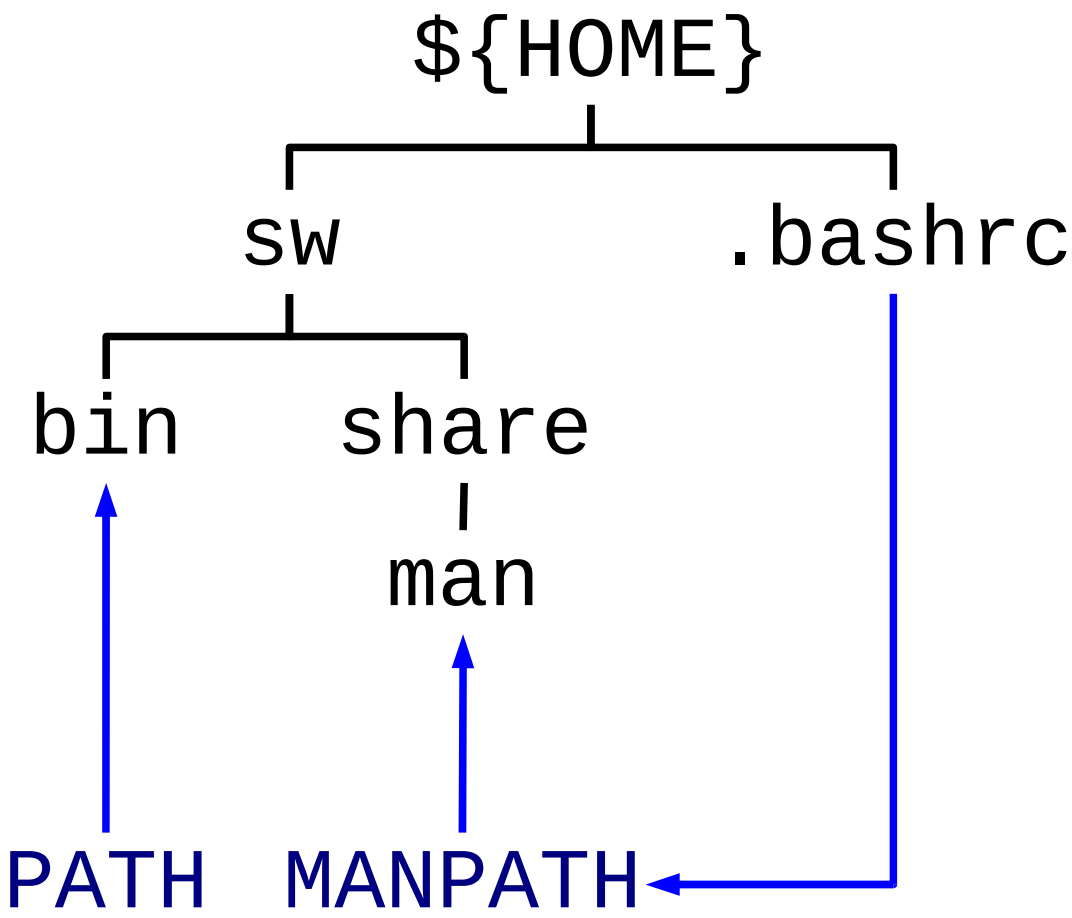
3. In your existing terminal window…

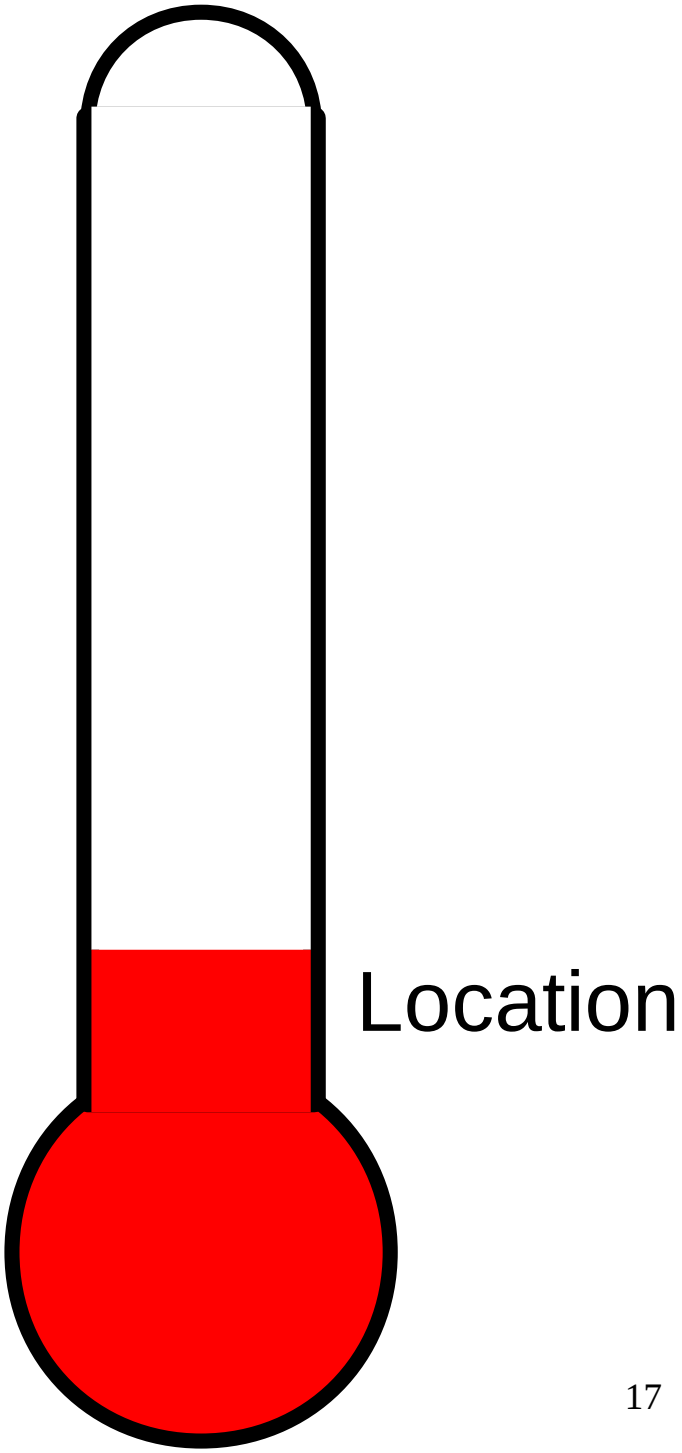
```
$ hello
-bash: hello: command not found
```

4. Launch and use a new terminal window…

```
$ hello
Hello, world!
```

5. Close the old terminal window.

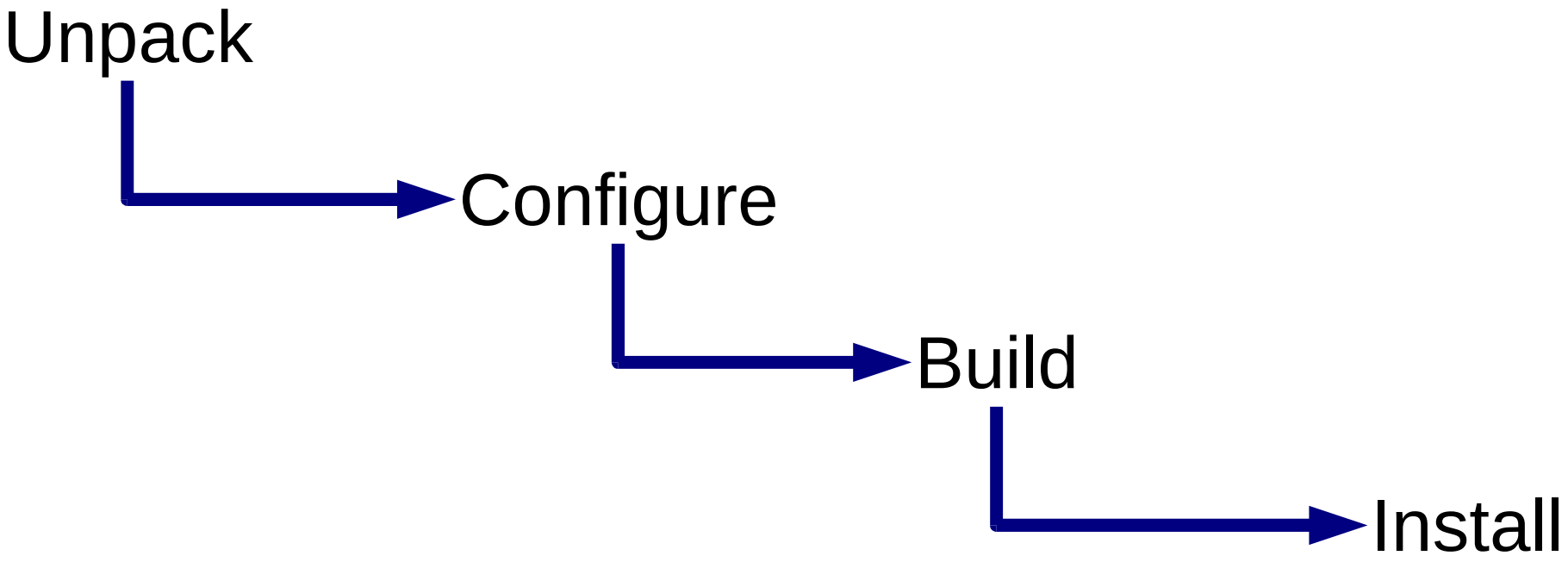

${HOME}
sw
.bashrc
bin
share
man
PATH
MANPATH


Location

We have a location…

…so let's build something to put in it!

The classic model

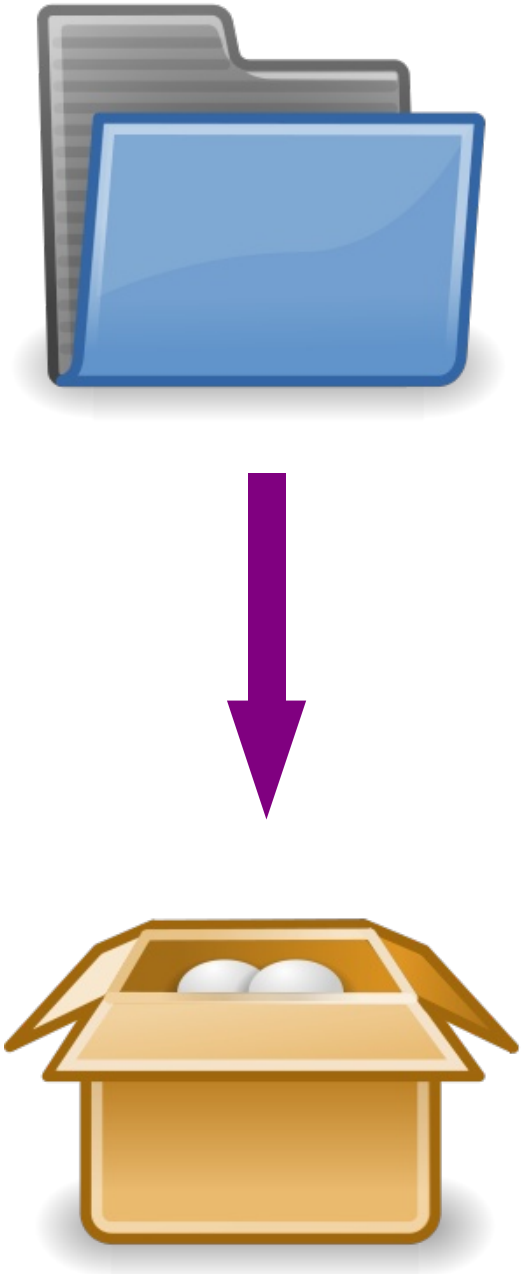


Unpack

Configure

Build

Install



`thing/`	directory
`thing.tar`	tar file
`thing.tar.Z`	compressed tar files
`thing.tar.gz`	compressed tar files
`thing.tgz`	compressed tar files
`thing.tar.bz2`	compressed tar files
`thing.zip`	zip file

`tar`: unpacking

tar -x -f thing.tgz

extract

from **f**ile

file name

`tar`: examining

```
thing/
thing/foo.c
thing/foo.h
thing/bar.c
thing/main.c
```

```
tar -t -f thing.tgz
```

table of contents

from **f**ile

file name

`zip`: unpacking

`unzip` thing.zip

file name

`zip`: examining

thing/
thing/foo.c
thing/foo.h
thing/bar.c
thing/main.c

unzip -t thing.zip

testing

file name

Worked example

1. prep

```
$ mkdir /tmp/building

$ cd /tmp/building

$ cp /ux/Lessons/Building/xdaliclock-2.20.tar.bz2 /tmp/building

$ ls
xdaliclock-2.20.tar.bz2
```

Worked example

2. unpacking

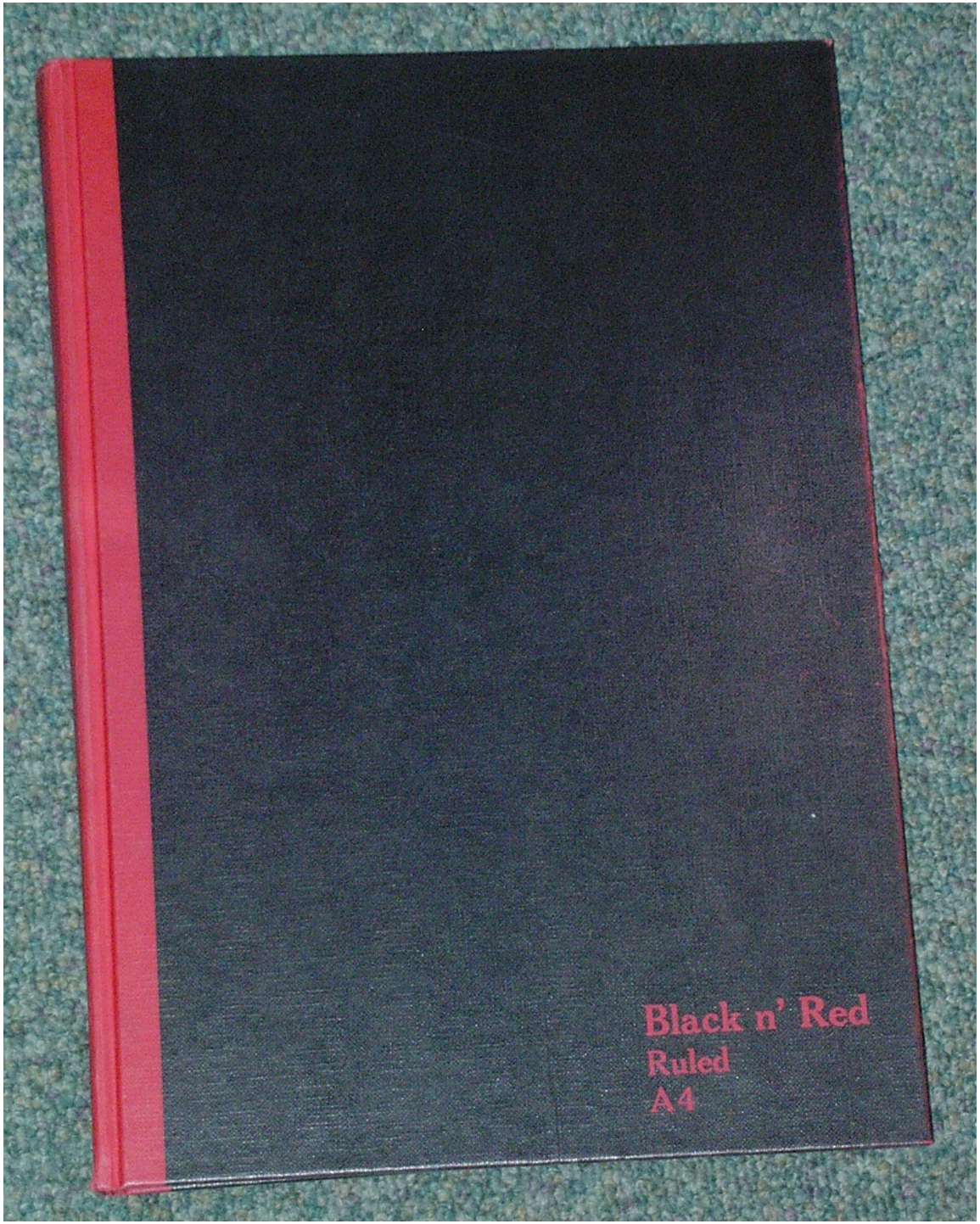
```
$ tar -x -f xdaliclock-2.20.tar.bz2

$ ls
xdaliclock-2.20  xdaliclock-2.20.tar.bz2
```

Keep records

Software:
version, source…

Details:
platform, options…

Results:
Success / Failure ?



Worked example

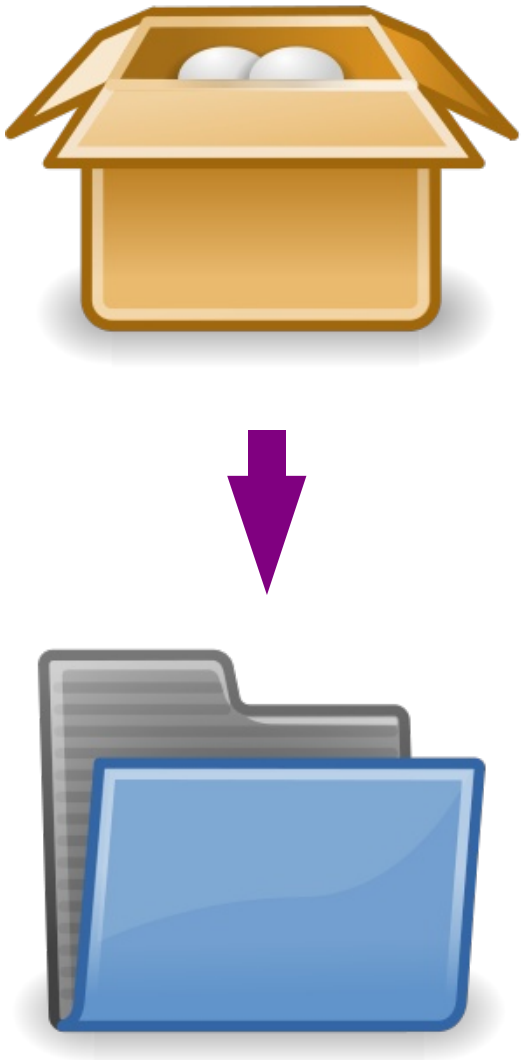
3. lab book

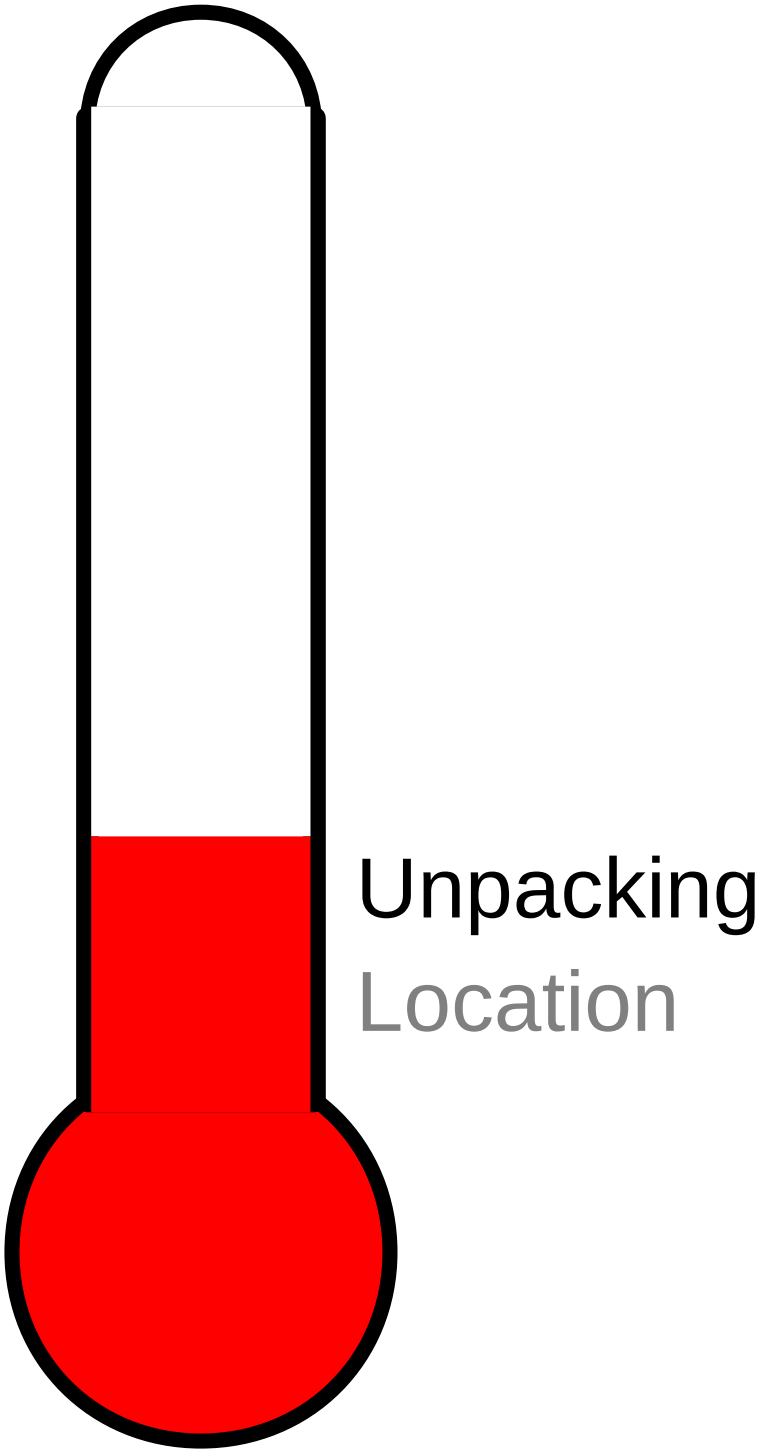
10th March 2008

xdaliclock v2.20

Source: UCS PWF Linux
/ux/Lessons/Building/xdaliclock-2.20.tar.bz2

Unpacks OK (tar -xf …)




Unpacking
Location

Coffee break

Five minutes break

Spines
Wrists
Eyes

Brains!

The README file

…

To build for the X Window System:

```
cd xdaliclock/X11/
./configure
make
make install
```

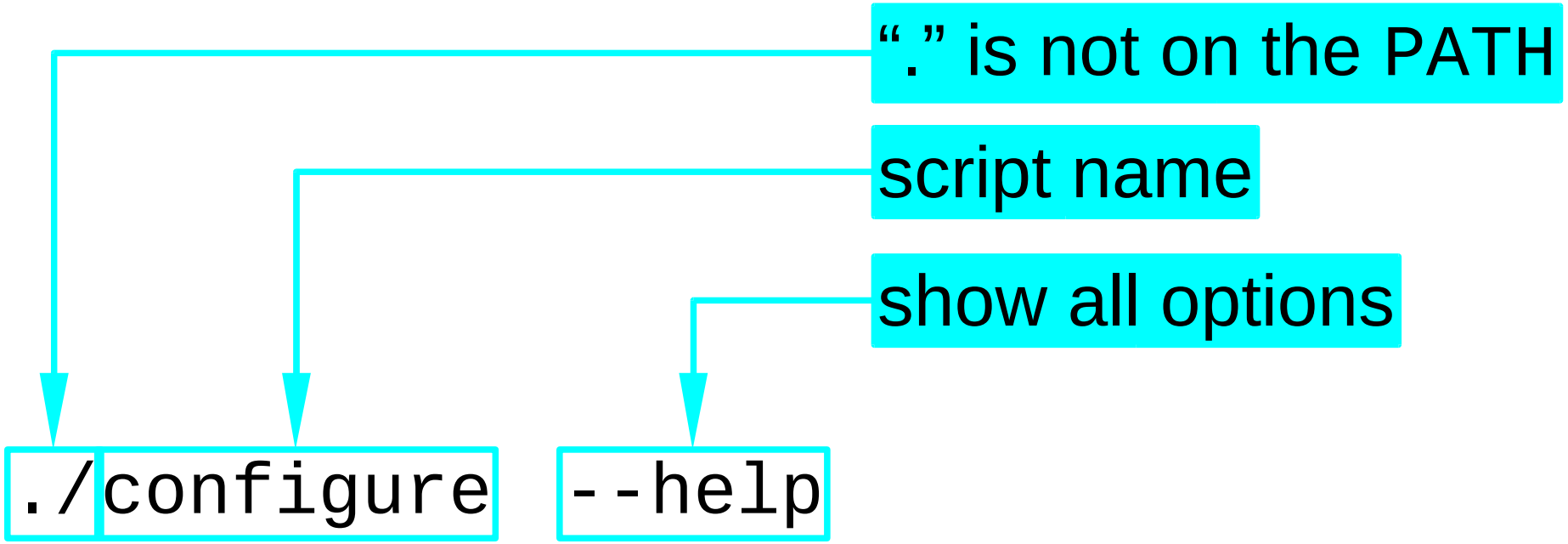
…

Unpack

Configure

Build

Install


“.” is not on the PATH
script name
show all options
./configure --help

```
./configure   --prefix="${HOME}/sw"
```

software location

our personal software directory

Compiler choice

```
./configure   --prefix=…    CC=gcc
```

specify
C compiler

Use the gcc
C compiler

Compiler options

CC	C compiler
CFLAGS	C compiler options
CXX	C++ compiler
CXXFLAGS	C++ compiler options
FC	Fortran compiler
FFLAGS	Fortran compiler options
LDFLAGS	Library options

Worked example

4. configuration

```
$ cd /tmp/building/xdaliclock-2.20
```

You are probably here already

```
$ cd X11
```

README's instructions

Configure for our location

```
$ ./configure --prefix="${HOME}/sw"
```

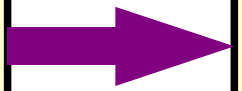
What `configure` does

…@prefix@…

…@CC@…

…@UIC@…

`Makefile.in`



…/home/rjd4/sw…

…gcc…

…/usr/lib/qt3/bin/uic…

`Makefile`

Worked example

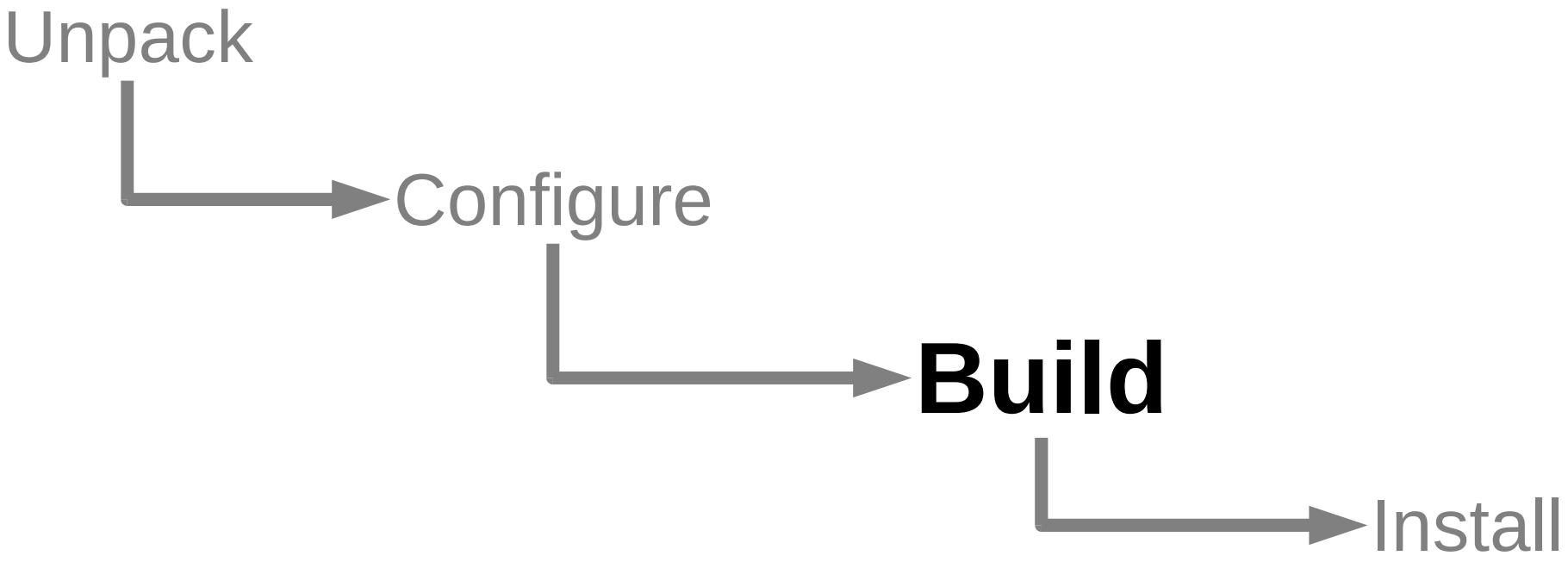
5. lab book

Source: UCS PWF Linux
/ux/Lessons/Building/xdaliclock-2.20.tar.bz2

Unpacks OK (tar -xf …)

```
./configure --prefix="${HOME}/sw"
```

Configures OK.


Unpack
Configure
Build
Install

make

`fubar.c` → `fubar.o`

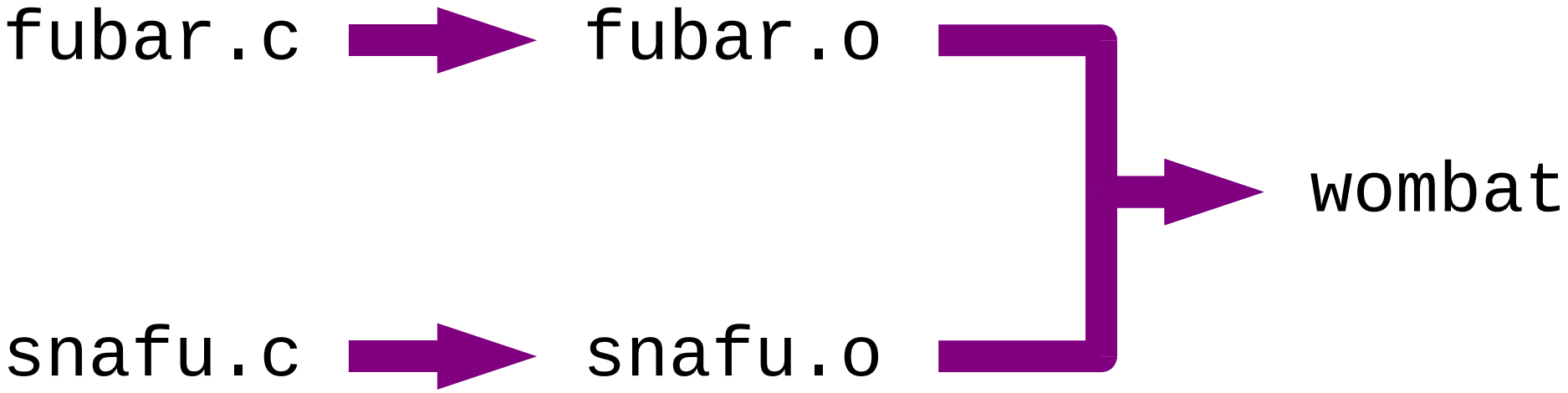
If:

exists missing

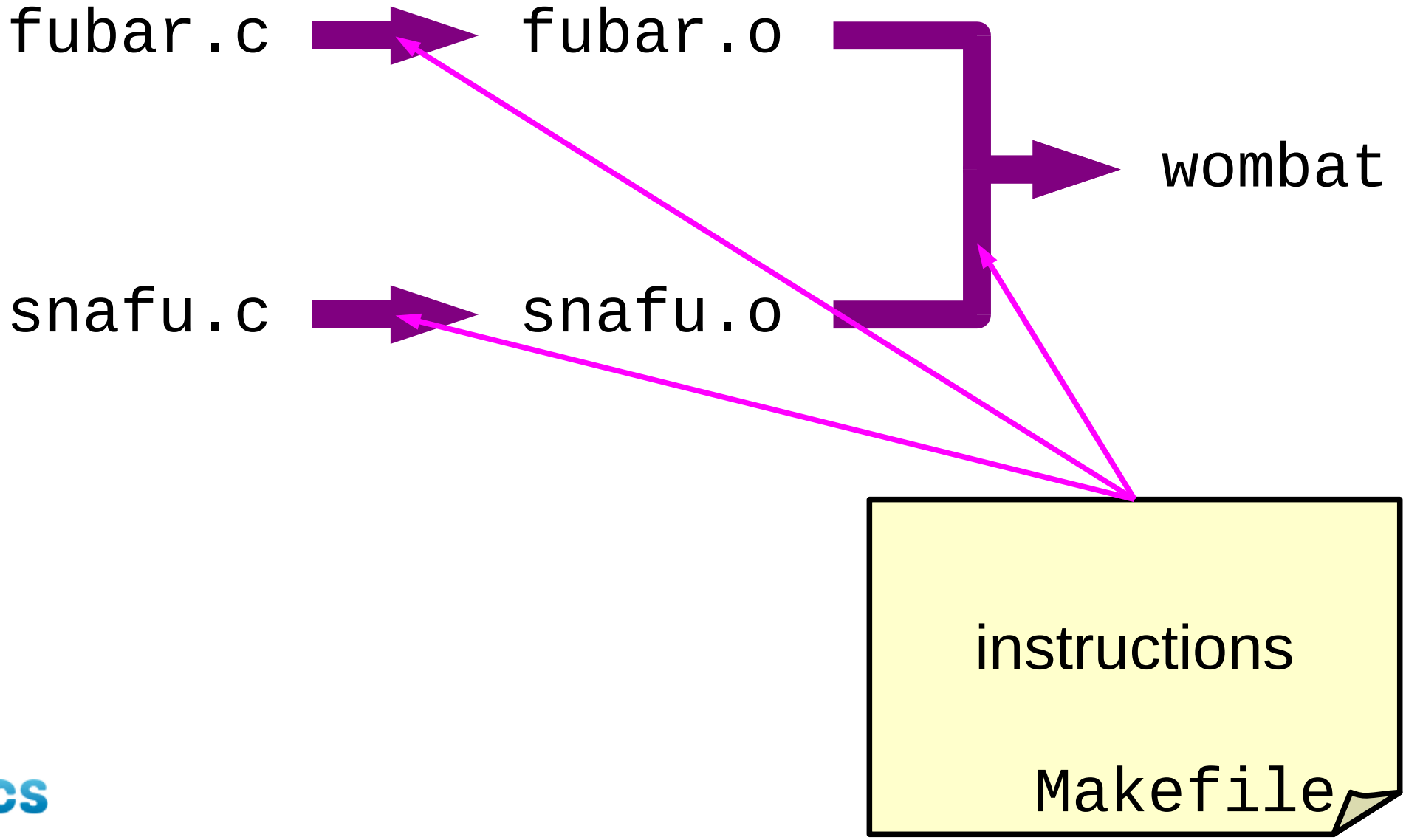
or:

newer older

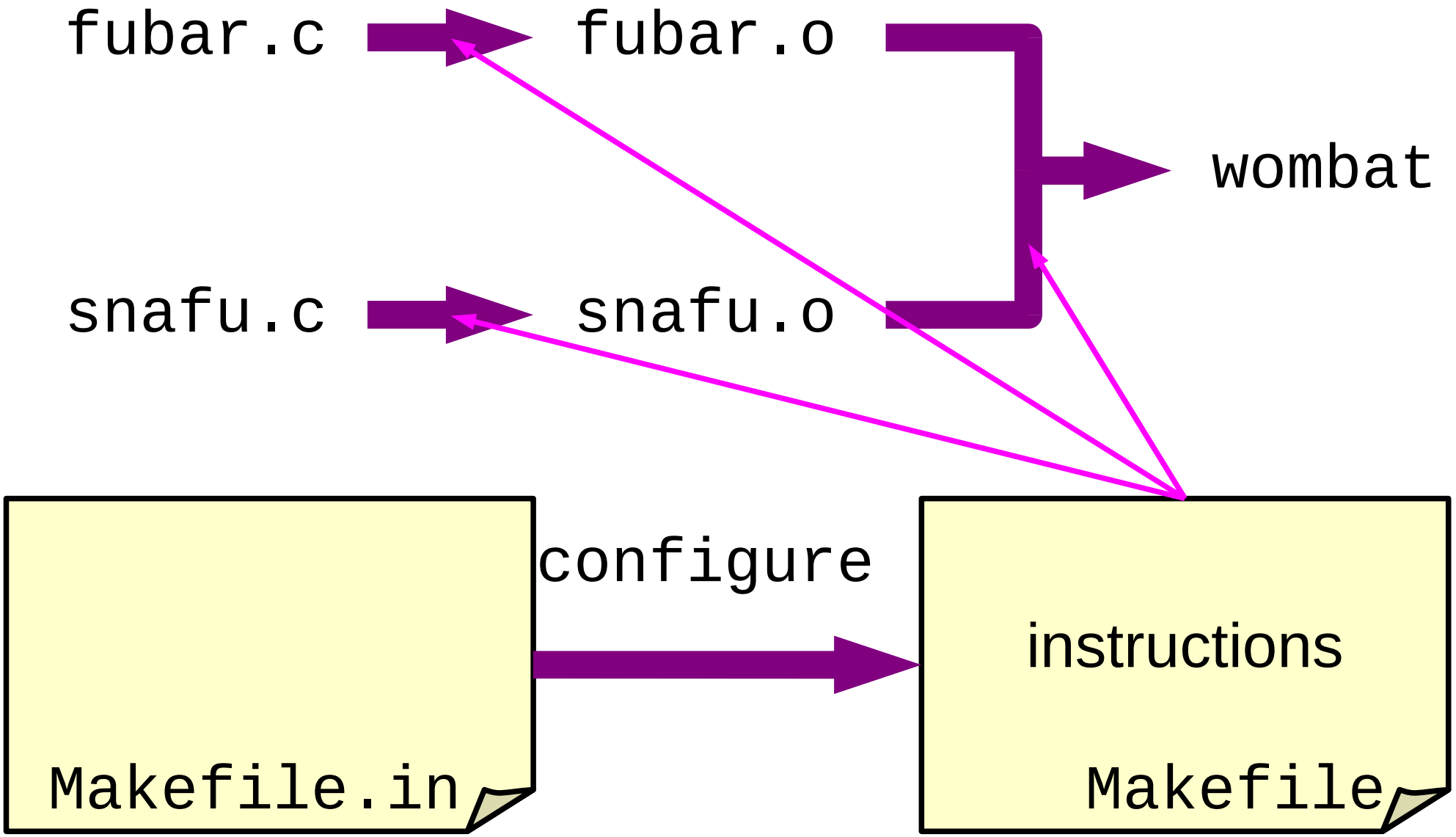
make



make


fubar.c
fubar.o
wombat
snafu.c
snafu.o
instructions
Makefile

make


fubar.c
fubar.o
wombat
snafu.c
snafu.o
configure
instructions
Makefile.in
Makefile

```
$ make
```

Worked example

6. make

```
config.status: creating Makefile
config.status: creating config.h

$ make
gcc -Wall -Wstrict-prototypes
-Wnested-externs -std=c89
-U__STRICT_ANSI__ -c -I. -I. -I./..
-I/home/rjd4/sw/include -DHAVE_CONFIG_H
-g -O2  xdaliclock.c
…
```

Worked example

7. confirmation

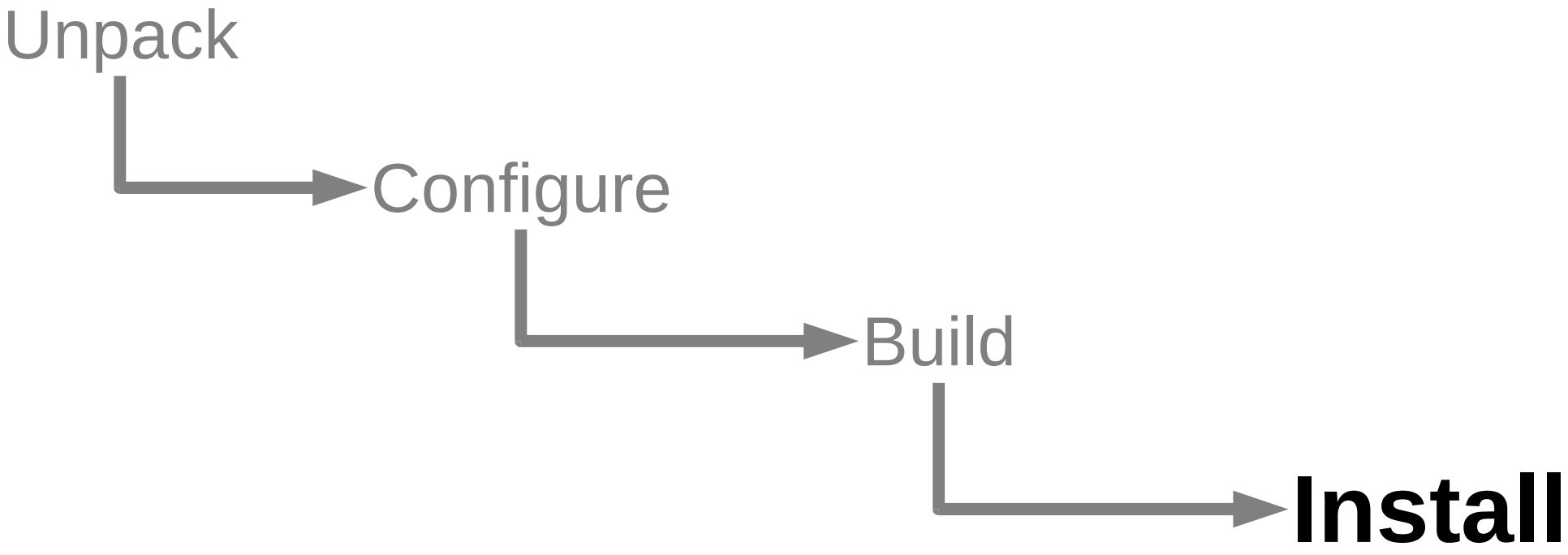
```
$ ls -l xdaliclock
-rwxr-xr-x        …      xdaliclock
```

Worked example

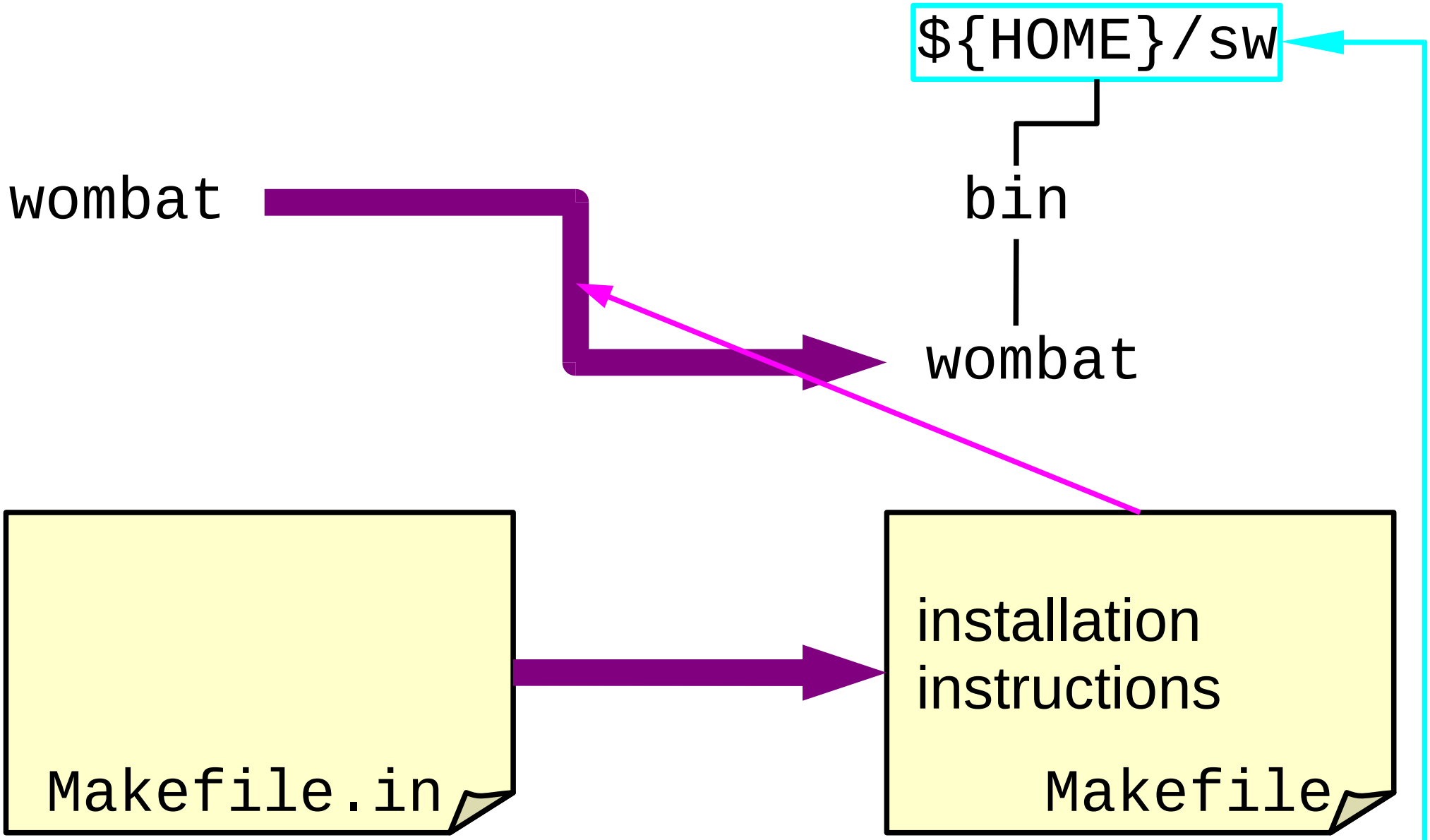
8. lab book

./configure –prefix="${HOME}/sw"
Configures OK.

make
Builds OK.


Unpack
Configure
Build
Install

make install




```
configure --prefix="${HOME}/sw"
```

```
$ make install
```

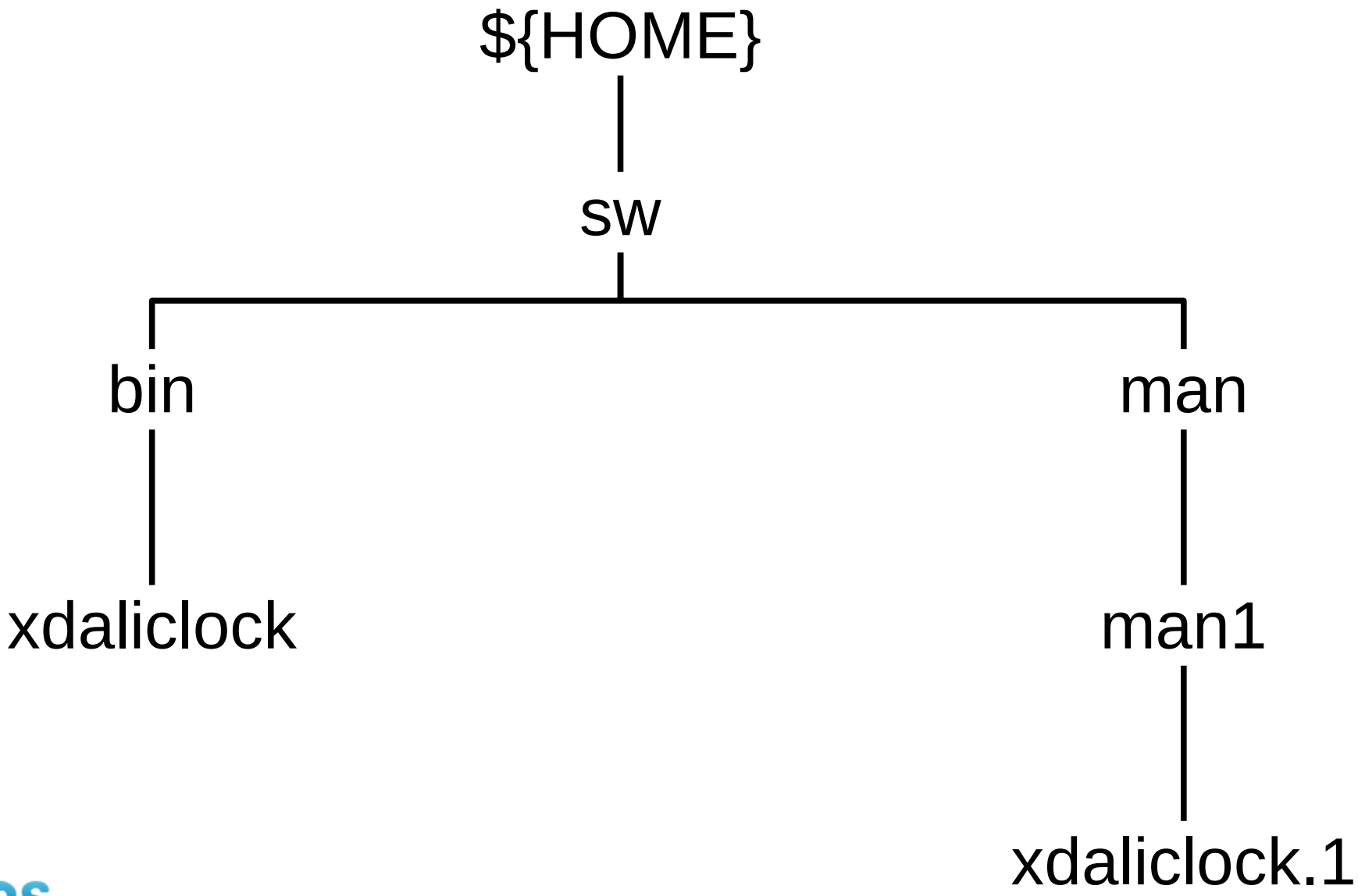
Worked example

9. installation

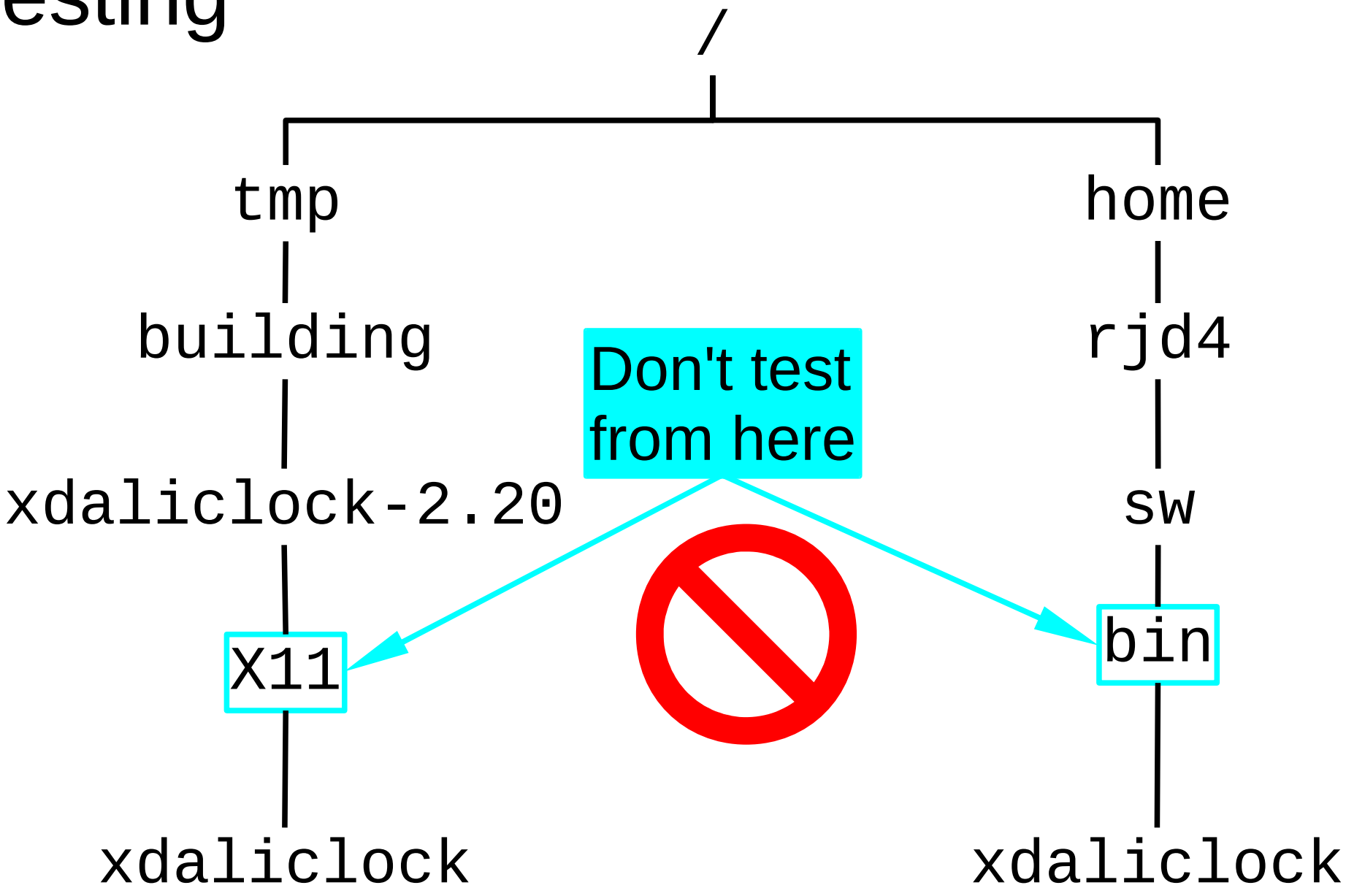
```
$ make install

install -c xdaliclock /home/rjd4/sw/
bin/xdaliclock

install -c ./xdaliclock.man /home/rj
d4/sw/man/man1/xdaliclock.1
```

Worked example

9. installation



Testing


/
tmp
home
building
rjd4
Don't test from here
xdaliclock-2.20
sw
X11
bin
xdaliclock
xdaliclock

Worked example

10. testing

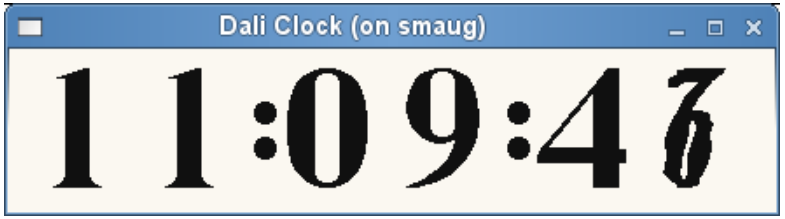
```
$ cd
```

go to home directory

```
$ xdaliclock  &
```

new command

run in background



Worked example

11. testing

```
$ type xdaliclock
xdaliclock is /home/rjd4/sw/bin/xdaliclock
```

Worked example

12. lab book

make
Builds OK.

make install
Installs OK.
Works from home directory.

Long builds & installs

```
make
```

```
make install
```

```
make && make install
```

First

and if it works

Second

Exercise

openbabel

`/ux/Lessons/Building` → `/tmp/building`

`openbabel-2.2.3.tar.gz`

1. unpack
2. configure
3. build
4. install

} &&

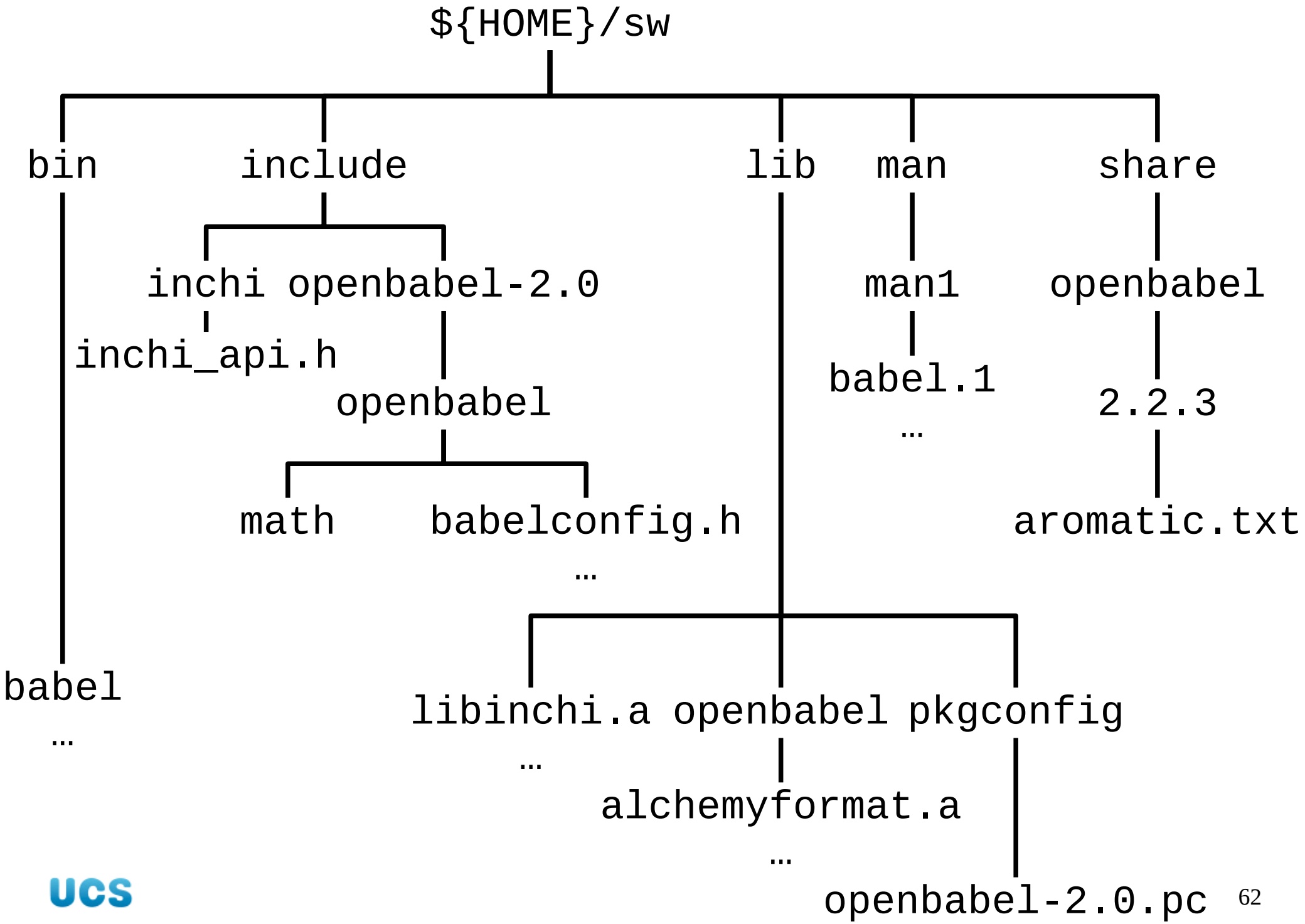
Lab book!

Coffee break

Ten minutes

Don't just stare at the screen!

UCS


${HOME}/sw
bin
include
lib
man
share
inchi
openbabel-2.0
man1
openbabel
inchi_api.h
babel.1
openbabel
2.2.3
…
math
babelconfig.h
aromatic.txt
…
babel
…
libinchi.a
openbabel
pkgconfig
…
alchemyformat.a
…
openbabel-2.0.pc

Exercise

openbabel

/ux/Lessons/Building → /tmp/building

ethanol.cml

scene.pov

```
$ babel ethanol.cml ethanol.xyz
```

format conversion

Exercise

libghemical
liboglappth

/ux/Lessons/Building → /tmp/building

libghemical-2.99.2.tar.gz
liboglappth-0.98.tar.gz

Dependencies

ghemical

needs

openbabel
libghemical
liboglappth

needs

base system

Worked example

`/ux/Lessons/Building` → `/tmp/building`

`ghemical-2.99.2.tar.gz`

Failed dependency

```
$ ./configure --prefix="${HOME}/sw"
…No package 'openbabel-2.0' found…
```

✘

pkg-config

PKG_CONFIG_PATH

?

pkg-config

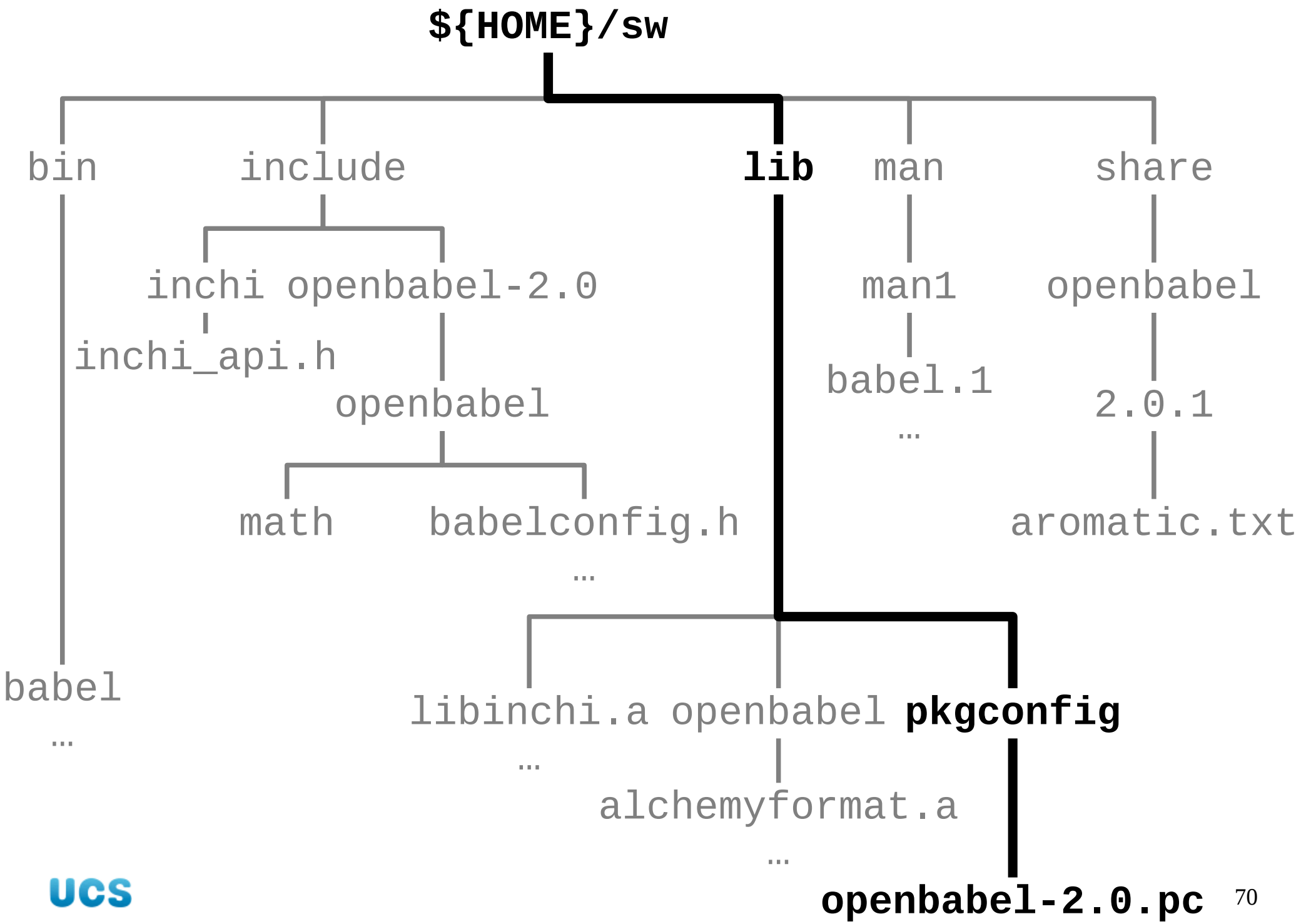
What are the library options for…

```
$ pkg-config --libs gtkglext-1.0
-Wl,--export-dynamic -lgtkglext-x11-1.0
-lgdkglext-x11-1.0 -lGLU -lGL -lXmu
-lXt -lSM -lICE -lgtk-x11-2.0
-lpangox-1.0 -lX11 -lgdk-x11-2.0
-latk-1.0 -lgio-2.0 -lpangoft2-1.0
-lgdk_pixbuf-2.0 -lpangocairo-1.0
-lcairo -lpango-1.0 -lfreetype -lz
-lfontconfig -lgobject-2.0 -lgmodule-2.0
-lglib-2.0
```

pkg-config

Our software

```
$ pkg-config --libs openbabel-2.0

pkg-config --libs openbabel-2.0
Package openbabel-2.0 was not found
in the pkg-config search path.
Perhaps you should add the directory
containing `openbabel-2.0.pc' to the
PKG_CONFIG_PATH environment variable.
No package 'openbabel-2.0' found
```


${HOME}/sw
bin
include
lib
man
share
inchi
openbabel-2.0
man1
openbabel
inchi_api.h
babel.1
…
2.0.1
openbabel
math
babelconfig.h
…
aromatic.txt
babel
…
libinchi.a
…
openbabel
pkgconfig
alchemyformat.a
…
openbabel-2.0.pc

${HOME}/.bashrc

Set this environment variable to be…

```
export PKG_CONFIG_PATH="
${HOME}/sw/lib/pkgconfig
:
${PKG_CONFIG_PATH} "
```

…our new directory…

…a colon…

…the old value

Exercise

1. Copy in a new ${HOME}/.bashrc file.

```
/ux/Lessons/Building/bashrc2
                          └──→${HOME}/.bashrc
```

2. In your existing terminal window…

```
$ pkg-config --libs openbabel-2.0
Package openbabel-2.0 was not found
in the pkg-config search path.
…
```

Exercise

3. Launch and use a new terminal window…

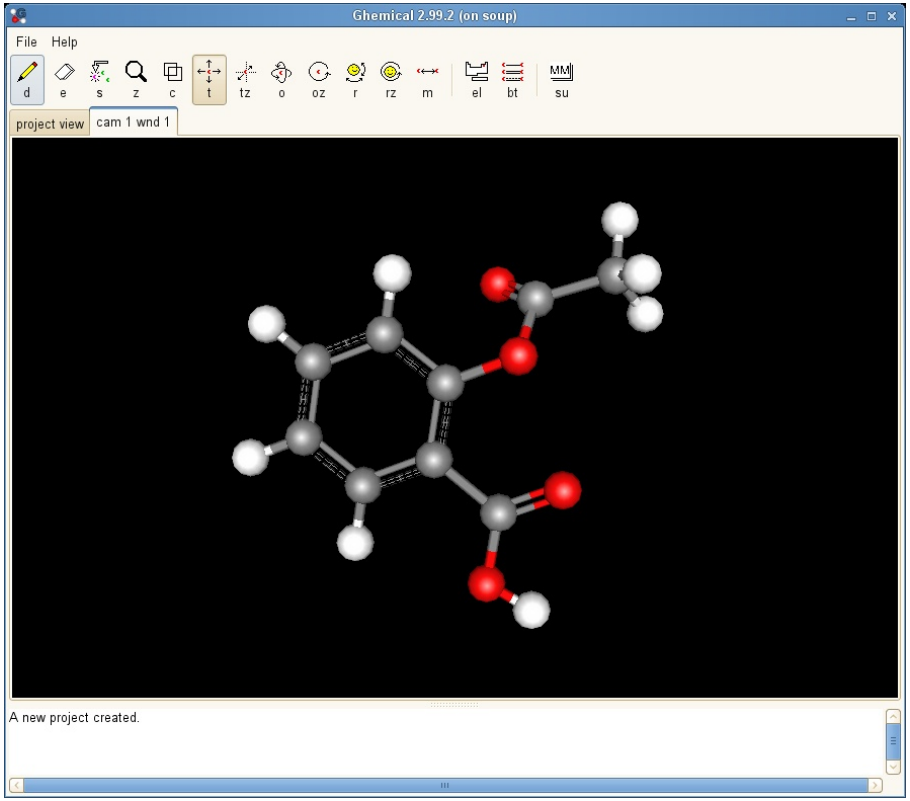
```
$ pkg-config --libs openbabel-2.0
-L/home/rjd4/sw/lib -lopenbabel
```

4. Close the old terminal window.

Exercise

`ghemical`

1. configure
2. build
3. install
4. launch

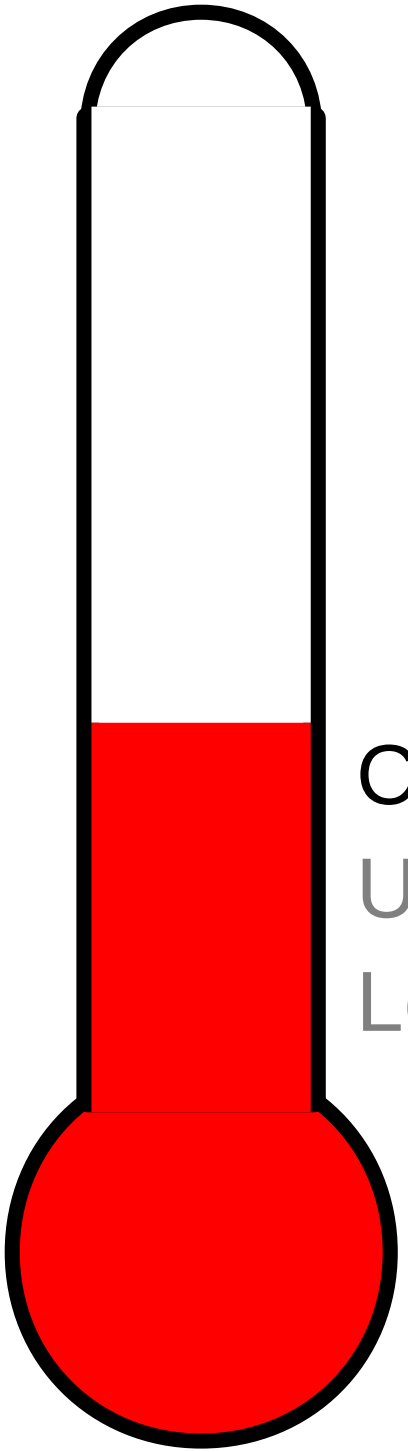


```
./configure
  --prefix=…
```

```
make
```

```
make install
```

```
PKG_CONFIG_PATH
```



Configured builds

Unpacking

Location