



Automated Builds

Macros

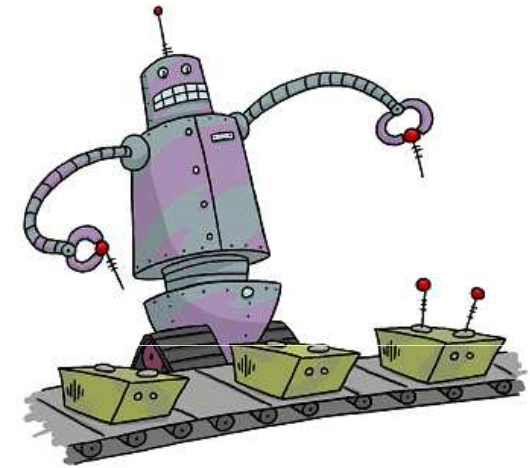


Copyright © Software Carpentry 2010

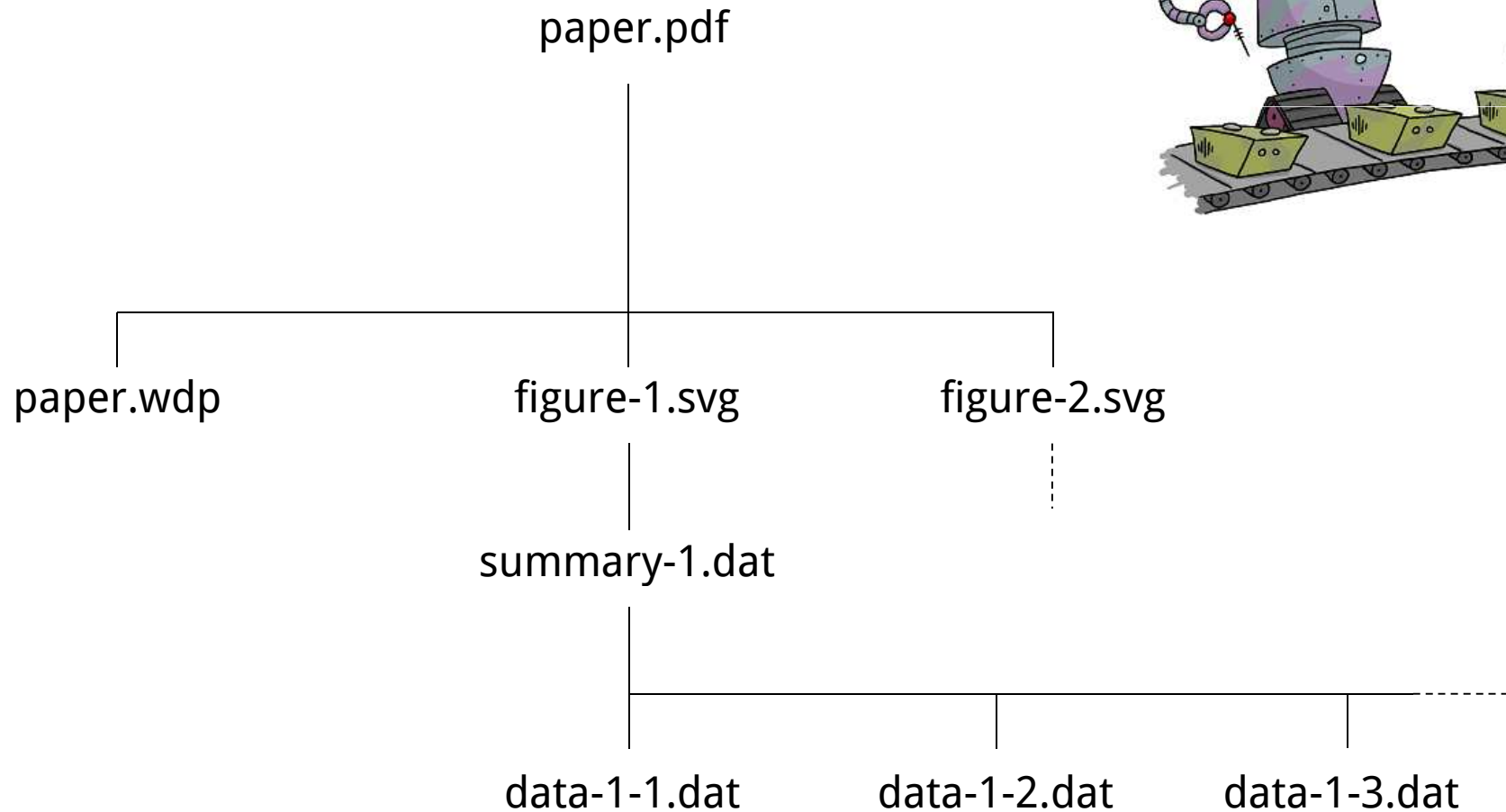
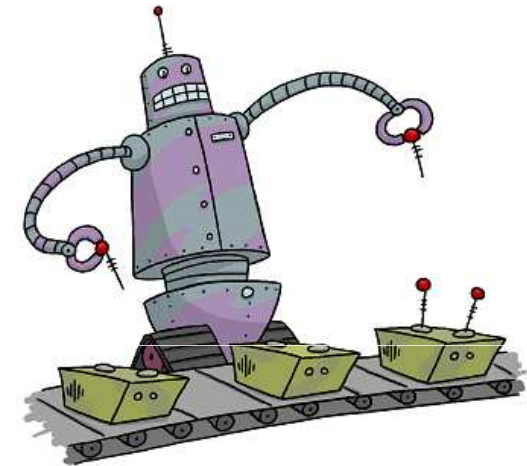
This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.

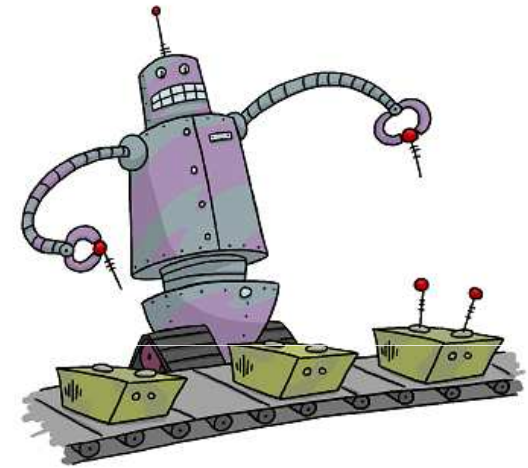
Manage tasks and dependencies



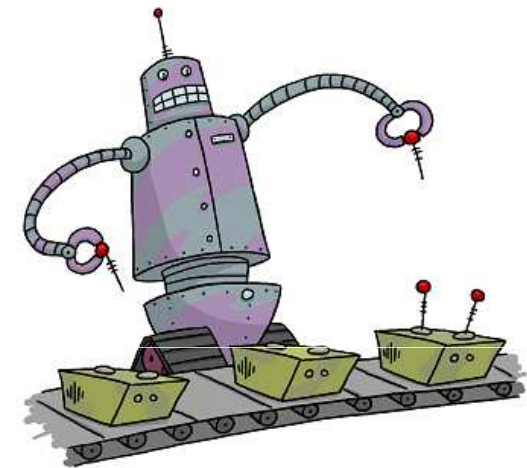
Manage tasks and dependencies



"must conform to university style"



"must conform to university style"



paper.pdf

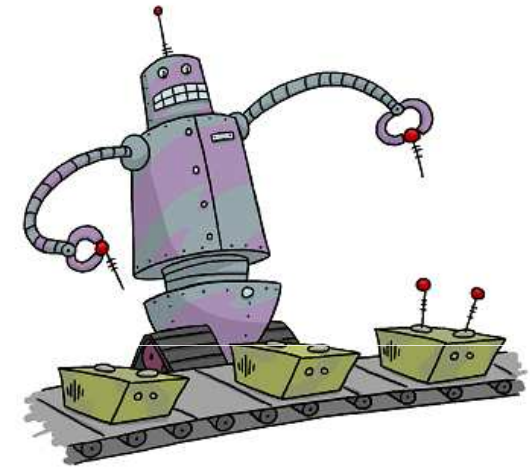
euphoric.wps

paper.wdp

figure-1.svg

figure-2.svg

"must conform to university style"



paper.pdf

euphoric.wps

paper.wdp

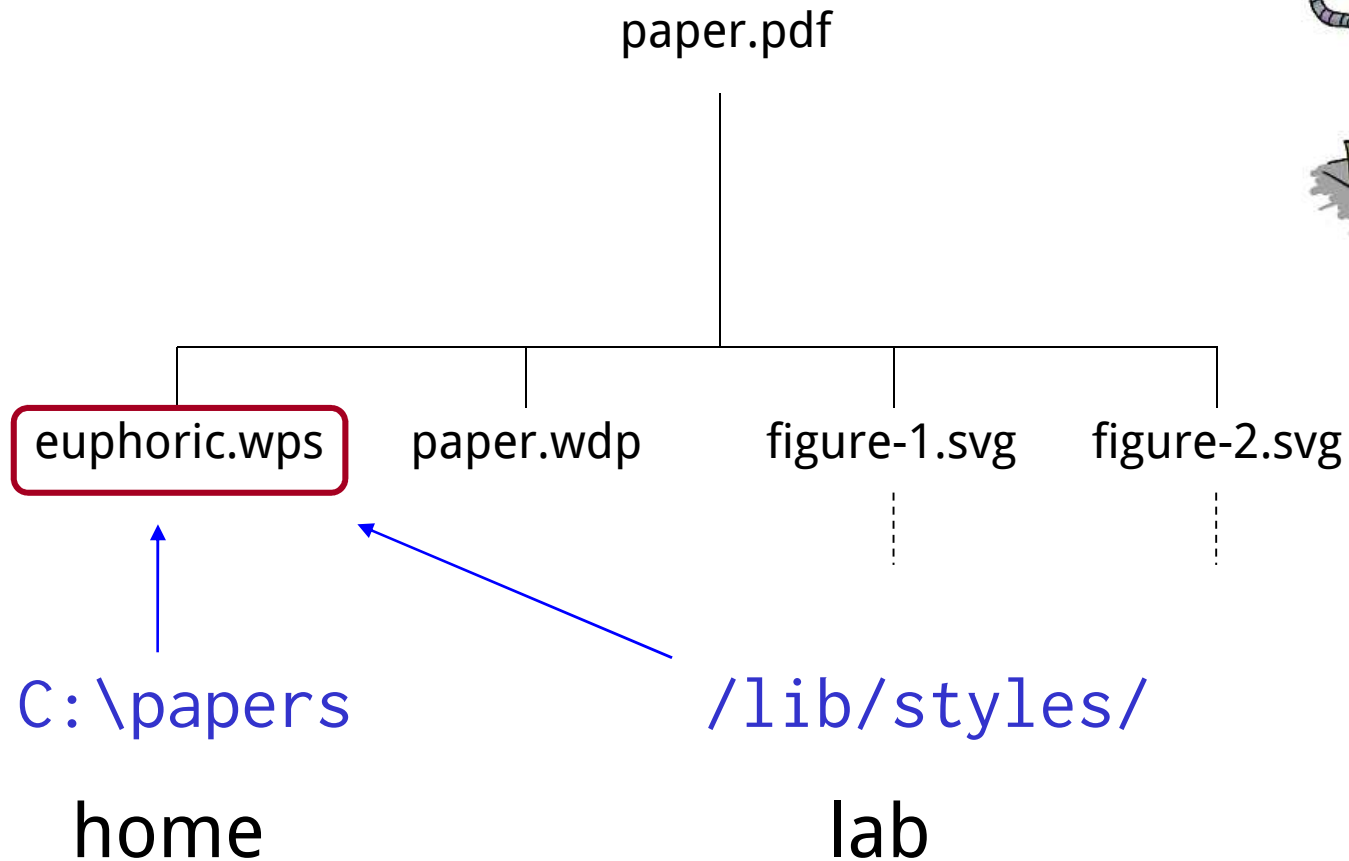
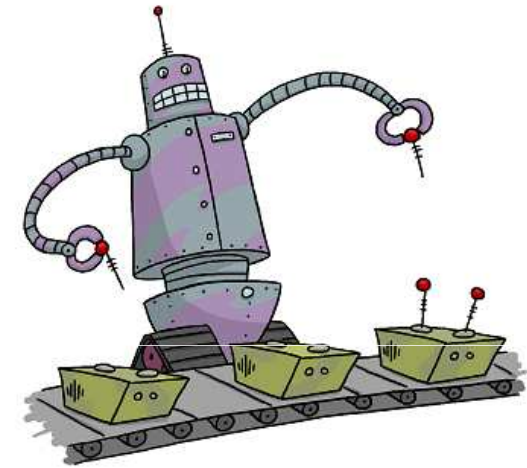
figure-1.svg

figure-2.svg

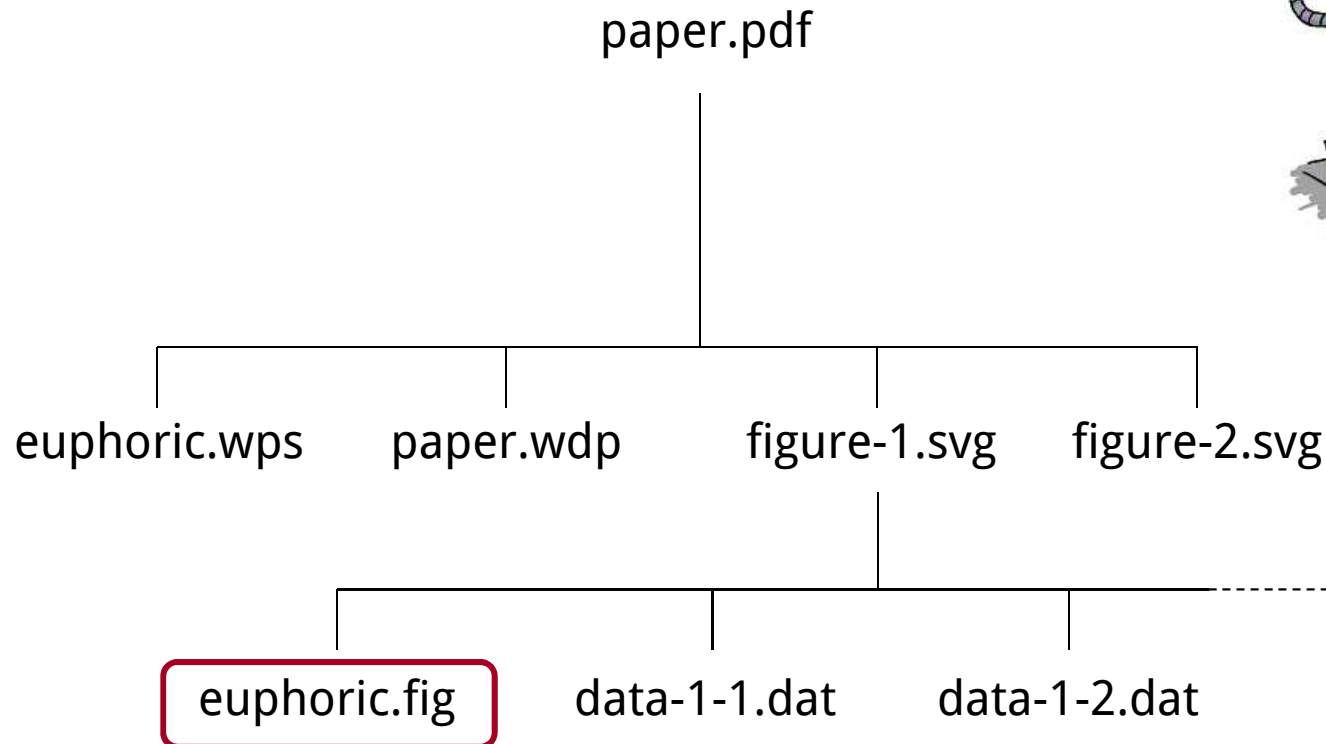
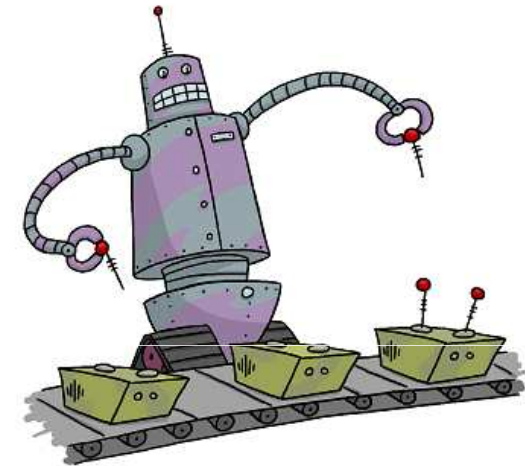
C:\papers

home

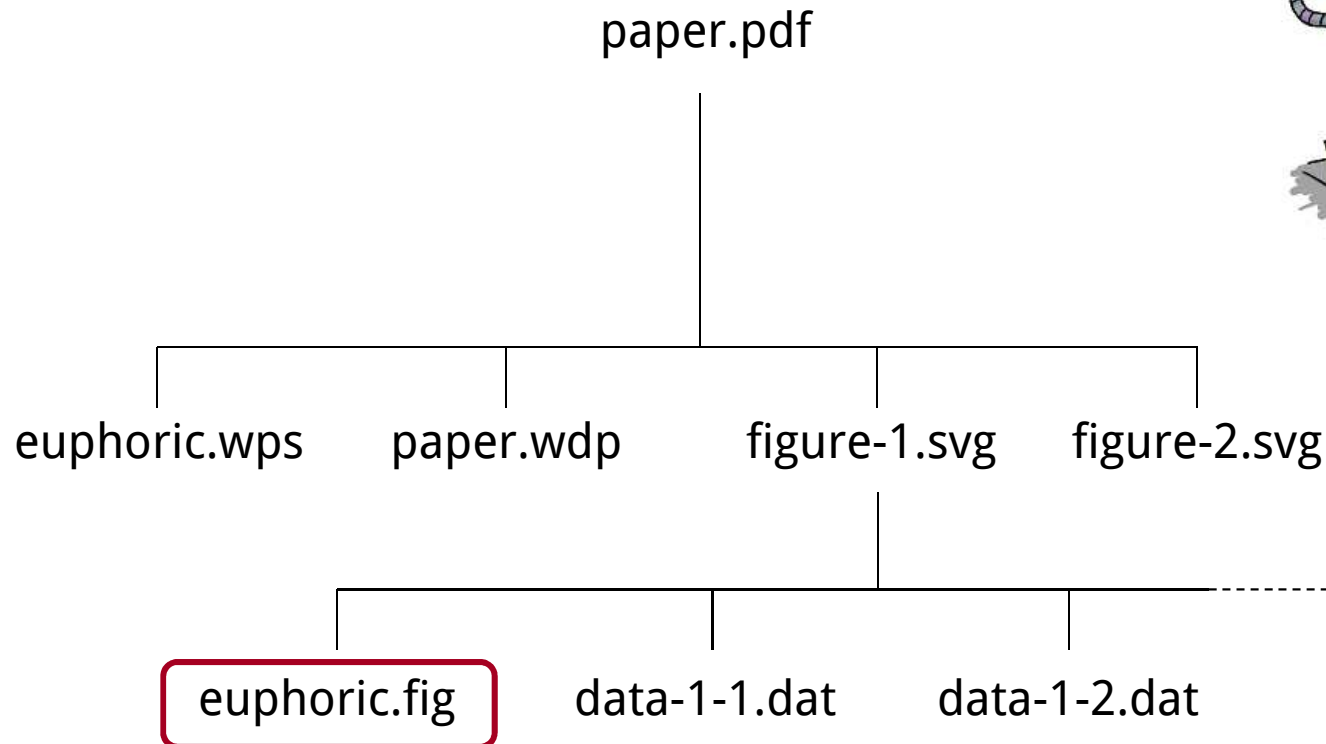
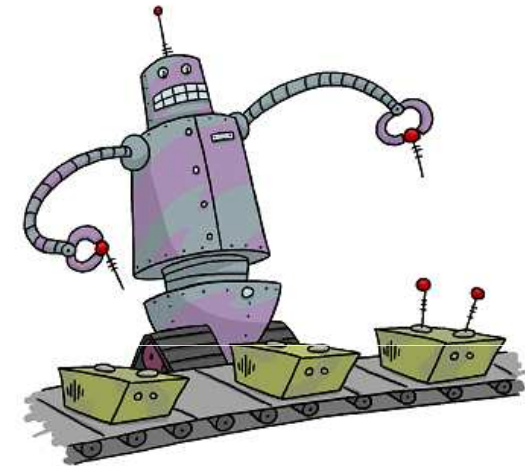
"must conform to university style"



"must conform to university style"



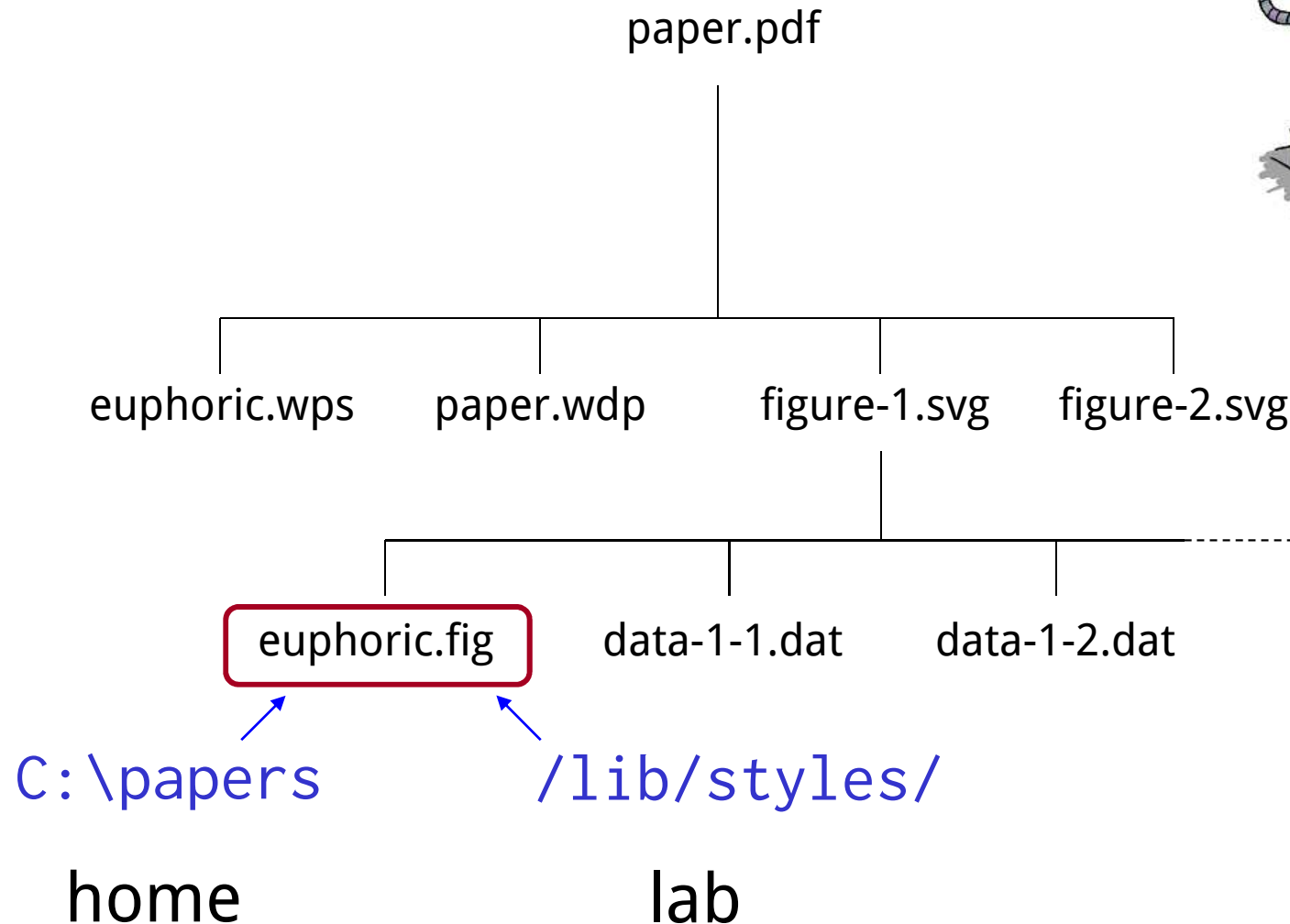
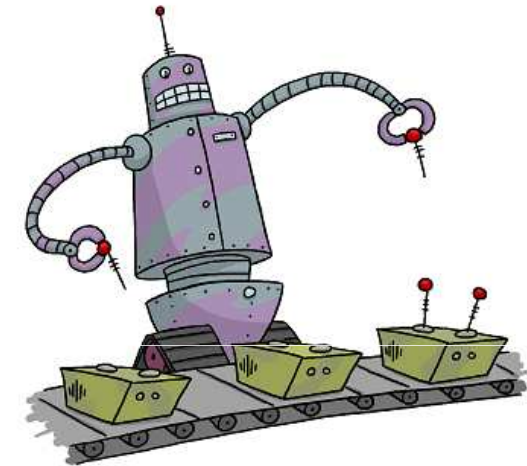
"must conform to university style"



C:\papers

home

"must conform to university style"



Makefile so far

```
# false-dependencies.mk

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf $<

figure-%.svg : summary-%.dat
    sgr -N -r $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-.*.dat : stats.py
    touch $@
```

Add directories for working at home

```
# with-directories-at-home.mk

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style c:/papers/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s c:/papers/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-.*.dat : stats.py
    touch $@
```

Add directories for working at home

```
# with-directories-at-home.mk

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style c:/papers/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s c:/papers/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-.*.dat : stats.py
    touch $@
```

Add directories for working at home

```
# with-directories-at-home.mk

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style c:/papers/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s c:/papers/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-*.dat : stats.py
    touch $@
```

Usually don't list "system" files explicitly

Add directories for working at home

```
# with-directories-at-home.mk

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style c:/papers/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s c:/papers/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-%-*.dat : stats.py
    touch $@
```

Usually don't list "system" files explicitly

But what about the lab?

1. Write two Makefiles

1. ~~Write~~ two Makefiles

Write and maintain

1. ~~Write~~ two Makefiles

Write and maintain

2. Comment and uncomment lines

1. ~~Write~~ two Makefiles

Write and maintain

2. Comment and uncomment lines

Consistently every time

1. ~~Write~~ two Makefiles

Write and maintain

2. Comment and uncomment lines

Consistently every time

Will create lots of noise in
version control

1. ~~Write~~ two Makefiles

Write and maintain

2. Comment and uncomment lines

Consistently every time

Will create lots of noise in
version control

3. Refactor

Use a *macro*

Use a *macro*

```
# with-macro.mk

STYLE_DIR=c:/papers/

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style ${STYLE_DIR}/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s ${STYLE_DIR}/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-*.*.dat : stats.py
    touch $@
```

Use a *macro*

```
# with-macro.mk
```

```
STYLE_DIR=c:/papers/
```

```
paper.pdf : paper.wdp figure-1.svg figure-2.svg  
    wdp2pdf --style ${STYLE_DIR}/euphoric.wps $<
```

```
figure-%.svg : summary-%.dat  
    sgr -N -r -s ${STYLE_DIR}/euphoric.fig $@ $^
```

```
summary-%.dat : data-%-*.dat  
    stats.py $@ $^
```

```
data-*-*.*.dat : stats.py  
    touch $@
```


Use a *macro*

```
# with-macro.mk

STYLE_DIR=c:/papers/

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf --style ${STYLE_DIR}/euphoric.wps $<

figure-%.svg : summary-%.dat
    sgr -N -r -s ${STYLE_DIR}/euphoric.fig $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-.*.dat : stats.py
    touch $@
```

Only have one thing to change

Only have one thing to change

✓ Consistency

Only have one thing to change

✓ Consistency

✗ But still have noise

Only have one thing to change

✓ Consistency

✗ But still have noise

Must use `${MACRO}` or `$(MACRO)`, *not* `$MACRO`

Only have one thing to change

✓ Consistency

✗ But still have noise

Must use `${MACRO}` or `$(MACRO)`, *not* `$MACRO`

Make reads `$MACRO` is `($M)ACRO`

Only have one thing to change

✓ Consistency

✗ But still have noise

Must use `${MACRO}` or `$(MACRO)`, *not* `$MACRO`

Make reads `$MACRO` is `($M)ACRO`

Which is probably just "ACRO"

Only have one thing to change

✓ Consistency

✗ But still have noise

Must use `${MACRO}` or `$(MACRO)`, *not* `$MACRO`

Make reads `$MACRO` is `($M)ACRO`

Which is probably just "ACRO"

Which is probably not what you want

Only have one thing to change

✓ Consistency

✗ But still have noise

Must use `${MACRO}` or `$(MACRO)`, *not* `$MACRO`

Make reads `$MACRO` is `($M)ACRO`

Which is probably just "ACRO"

Which is probably not what you want

yet another legacy wart

Commonly define macros for control flags

Commonly define macros for control flags

```
# with-lots-of-macros.mk
```

```
STYLE_DIR=c:/papers/  
WDP2PDF_FLAGS=--style ${STYLE_DIR}/euphoric.wps  
SGR_FLAGS=-N -r -s ${STYLE_DIR}/euphoric.fig
```



```
paper.pdf : paper.wdp figure-1.svg figure-2.svg  
    wdp2pdf ${WDP2PDF_FLAGS} $<
```

```
figure-%.svg : summary-%.dat  
    sgr ${SGR_FLAGS} $@ $^
```

```
summary-%.dat : data-%-*.dat  
    stats.py $@ $^
```

```
data-*-.*.dat : stats.py  
    touch $@
```

Commonly define macros for control flags

```
# with-lots-of-macros.mk
```

```
STYLE_DIR=c:/papers/  
WDP2PDF_FLAGS=--style ${STYLE_DIR}/euphoric.wps  
SGR_FLAGS=-N -r -s ${STYLE_DIR}/euphoric.fig
```

```
paper.pdf : paper.wdp figure-1.svg figure-2.svg  
    wdp2pdf ${WDP2PDF_FLAGS} $<
```

```
figure-%.svg : summary-%.dat  
    sgr ${SGR_FLAGS} $@ $^
```

```
summary-%.dat : data-%-*.dat  
    stats.py $@ $^
```

```
data-*-.*.dat : stats.py  
    touch $@
```

Commonly define macros for control flags

```
# with-lots-of-macros.mk
```

```
STYLE_DIR=c:/papers/  
WDP2PDF_FLAGS=--style ${STYLE_DIR}/euphoric.wps  
SGR_FLAGS=-N -r -s ${STYLE_DIR}/euphoric.fig
```

```
paper.pdf : paper.wdp figure-1.svg figure-2.svg  
    wdp2pdf ${WDP2PDF_FLAGS} $<
```

```
figure-%.svg : summary-%.dat  
    sgr ${SGR_FLAGS} $@ $^
```

```
summary-%.dat : data-%-*.dat  
    stats.py $@ $^
```

```
data-*-*.dat : stats.py  
    touch $@
```

Now put the first macro in a separate file

```
# config.mk
```

```
STYLE_DIR=c:/papers/
```

And include it from the main file

```
# with-include.mk
```

```
include config.mk
```

```
WDP2PDF_FLAGS=--style ${STYLE_DIR}/euphoric.wps
```

```
SGR_FLAGS=-N -r -s ${STYLE_DIR}/euphoric.fig
```

```
paper.pdf : paper.wdp figure-1.svg figure-2.svg  
    wdp2pdf ${WDP2PDF_FLAGS} $<
```

```
figure-%.svg : summary-%.dat  
    sgr ${SGR_FLAGS} $@ $^
```

```
summary-%.dat : data-%-*.dat  
    stats.py $@ $^
```

```
data-*-.*.dat : stats.py  
    touch $@
```

And include it from the main file

```
# with-include.mk
include config.mk


WDP2PDF_FLAGS=--style ${STYLE_DIR}/euphoric.wps
SGR_FLAGS=-N -r -s ${STYLE_DIR}/euphoric.fig

paper.pdf : paper.wdp figure-1.svg figure-2.svg
    wdp2pdf ${WDP2PDF_FLAGS} $<

figure-%.svg : summary-%.dat
    sgr ${SGR_FLAGS} $@ $^

summary-%.dat : data-%-*.dat
    stats.py $@ $^

data-*-.*.dat : stats.py
    touch $@
```



Actually create *two* configuration files

Actually create *two* configuration files

```
# config-home.mk
```

```
STYLE_DIR=c:/papers/
```

Actually create *two* configuration files

```
# config-home.mk
```

```
STYLE_DIR=c:/papers/
```

```
# config-lab.mk
```

```
STYLE_DIR=/lib/styles
```

Actually create *two* configuration files

```
# config-home.mk
```

```
STYLE_DIR=c:/papers/
```

```
# config-lab.mk
```

```
STYLE_DIR=/lib/styles
```

These two files stay in version control

Actually create *two* configuration files

```
# config-home.mk
```

```
STYLE_DIR=c:/papers/
```

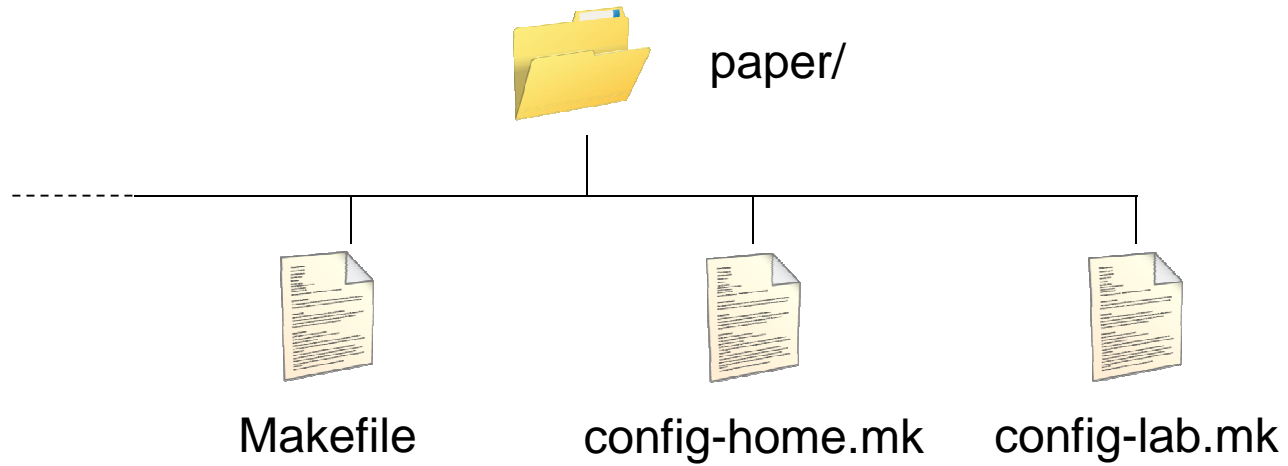
```
# config-lab.mk
```

```
STYLE_DIR=/lib/styles
```

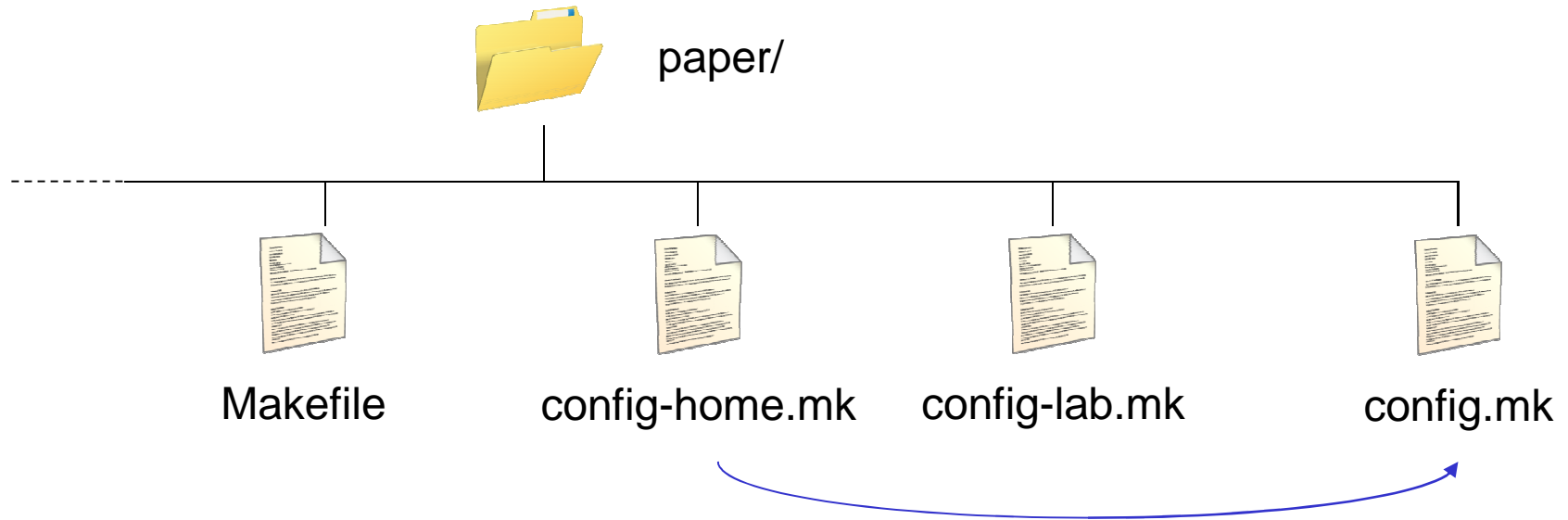
These two files stay in version control

Copy to create config.mk per machine

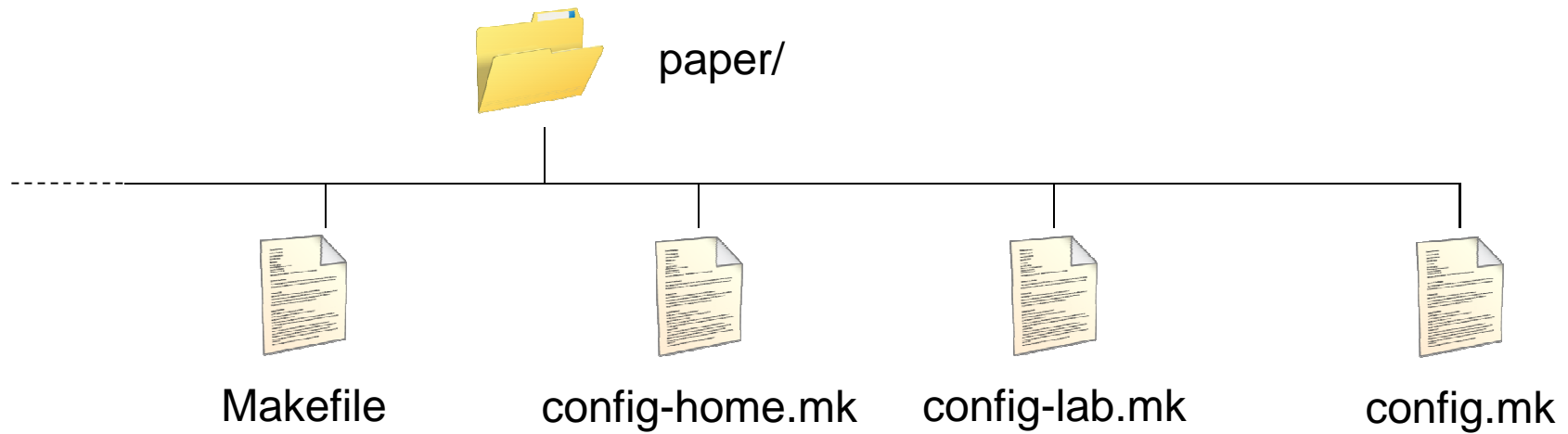
Home



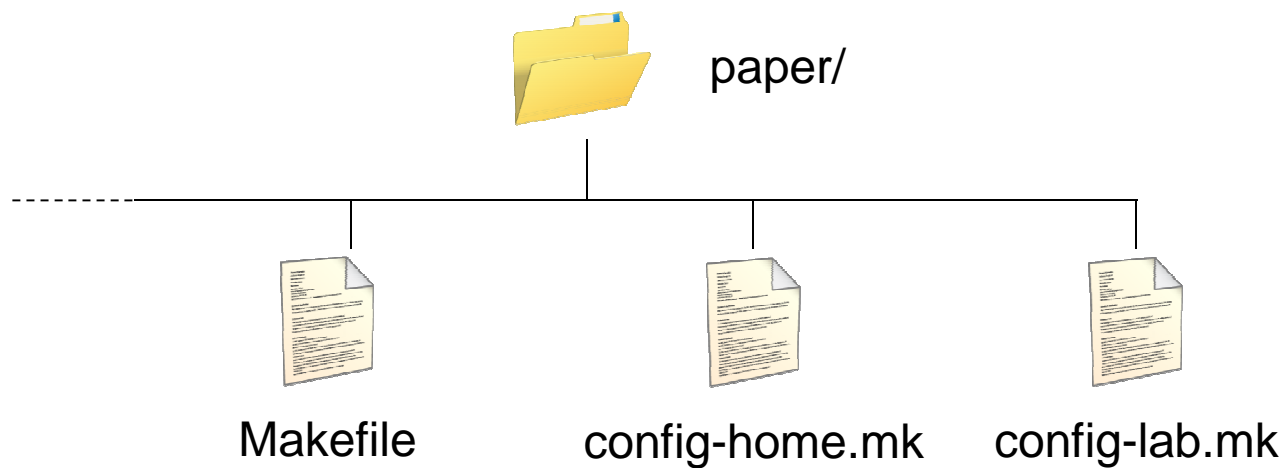
Home



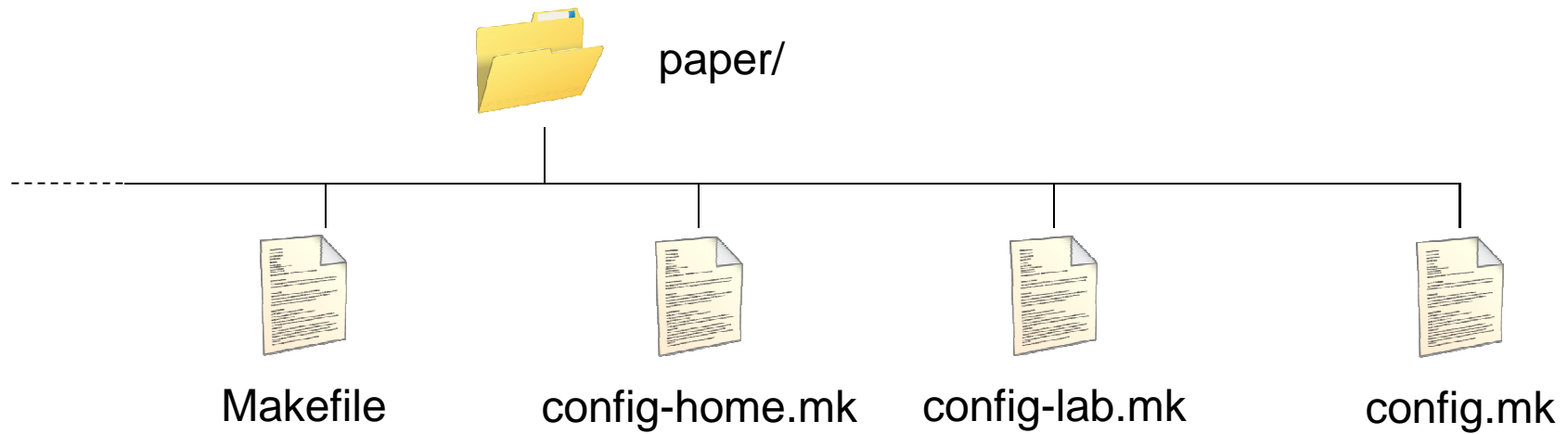
Home



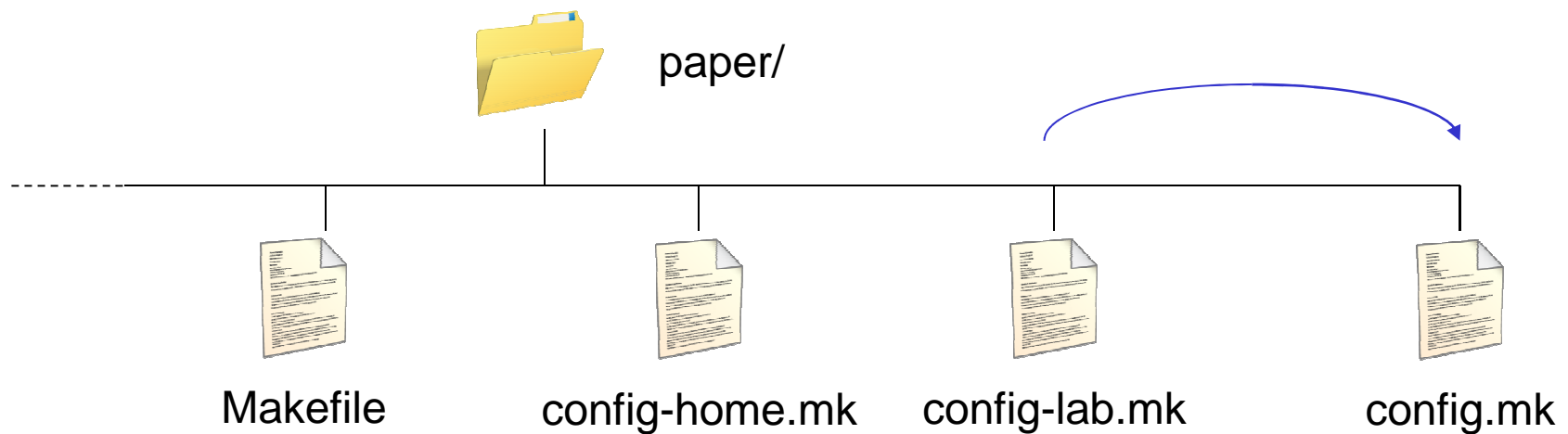
Lab



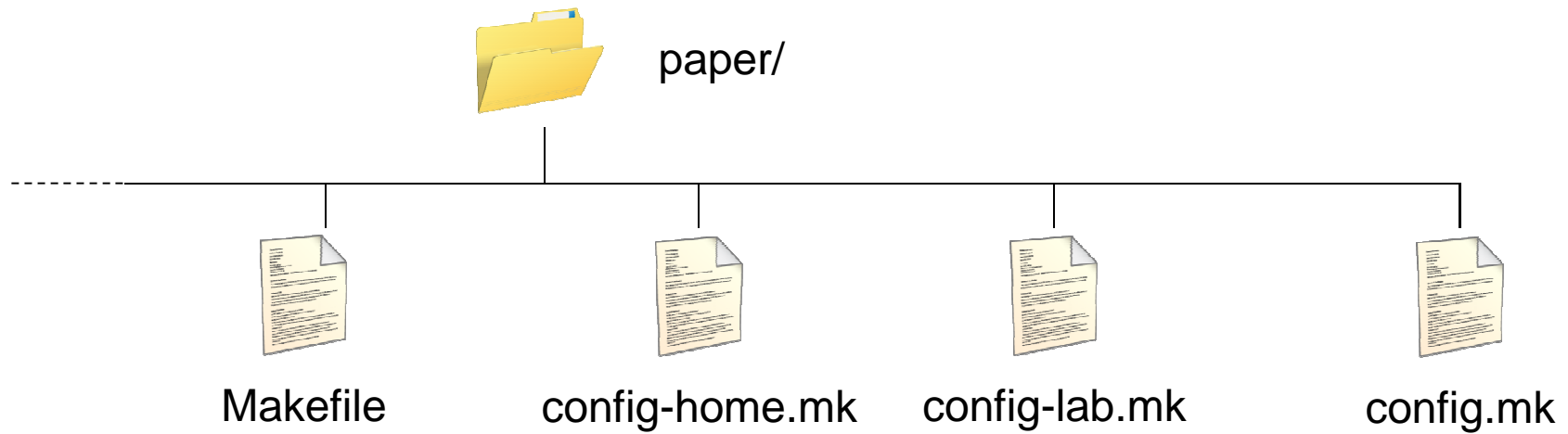
Home



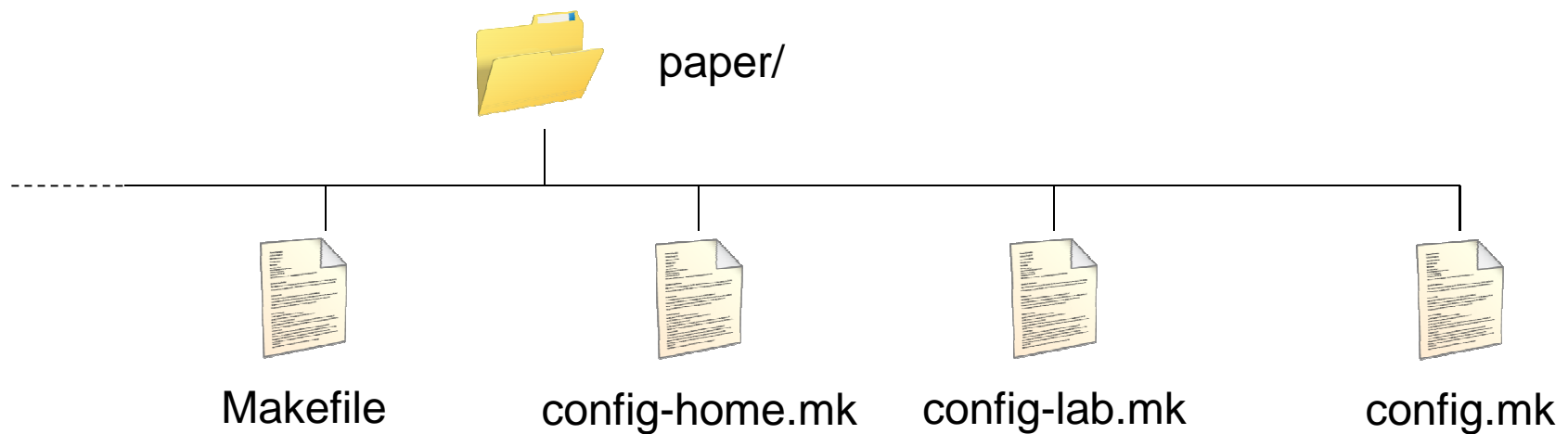
Lab



Home



Lab



Can also define macro value on command line

Can also define macro value on command line

```
$ make -DSTYLE_DIR=/lib/styles -f Makefile
```

Can also define macro value on command line

```
$ make -DSTYLE_DIR=/lib/styles -f Makefile
```

Generally a bad idea

Can also define macro value on command line

```
$ make -DSTYLE_DIR=/lib/styles -f Makefile
```

Generally a bad idea

Have to remember to type definition each time

Can also define macro value on command line

```
$ make -DSTYLE_DIR=/lib/styles -f Makefile
```

Generally a bad idea

Have to remember to type definition each time
correctly

Can also define macro value on command line

```
$ make -DSTYLE_DIR=/lib/styles -f Makefile
```

Generally a bad idea

Have to remember to type definition each time
correctly

And there's no record of the flag

Many other approaches

Many other approaches

CMake and Autoconf/Automake: compile higher-level specification into a Makefile (or equivalent)

Many other approaches

CMake and Autoconf/Automake: compile higher-level specification into a Makefile (or equivalent)

- ✓ Automatically discover/manage differences between machines

Many other approaches

CMake and Autoconf/Automake: compile higher-level specification into a Makefile (or equivalent)

✓ Automatically discover/manage differences between machines

✗ But even harder to debug

Many other approaches

CMake and Autoconf/Automake: compile higher-level specification into a Makefile (or equivalent)

✓ Automatically discover/manage differences between machines

✗ But even harder to debug

A build file is a program

Many other approaches

CMake and Autoconf/Automake: compile higher-level specification into a Makefile (or equivalent)

✓ Automatically discover/manage differences between machines

✗ But even harder to debug

A build file is a program

Requires the same degree of respect



created by

Greg Wilson

August 2010



Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.