

Installing common programs/applications on a research computer

Rob Porritt, G. G.
USArray short course, 2016

If on



first install the command line tools

From finder, open /Applications/Utilities/Terminal.app

Enter: `xcode-select --install`
and follow the graphical prompts

Then install XQuartz windowing system via .dmg from web

If on



then you should be good to go.

Check your distro for whether you should use
yum, apt-get, rpm, or something new to me

If on



look into Matlab or Python as cross-platform
working environments. You may need to
dual-boot with Linux or install cyg-win

Your computer needs a “path” variable.
This is a list of directories, separated by : which the
operating system searches for executable files

On the macs I set it by editing a file called:
~/.profile

and add lines in bash syntax such as:

```
export PATH=~/.bin/:${PATH}
```

Alternatively you could use ~/.bashrc

If using a tcsh shell, you'd edit your
~/.tcshrc

with lines like:

```
setenv PATH ~/.bin:${PATH}
```

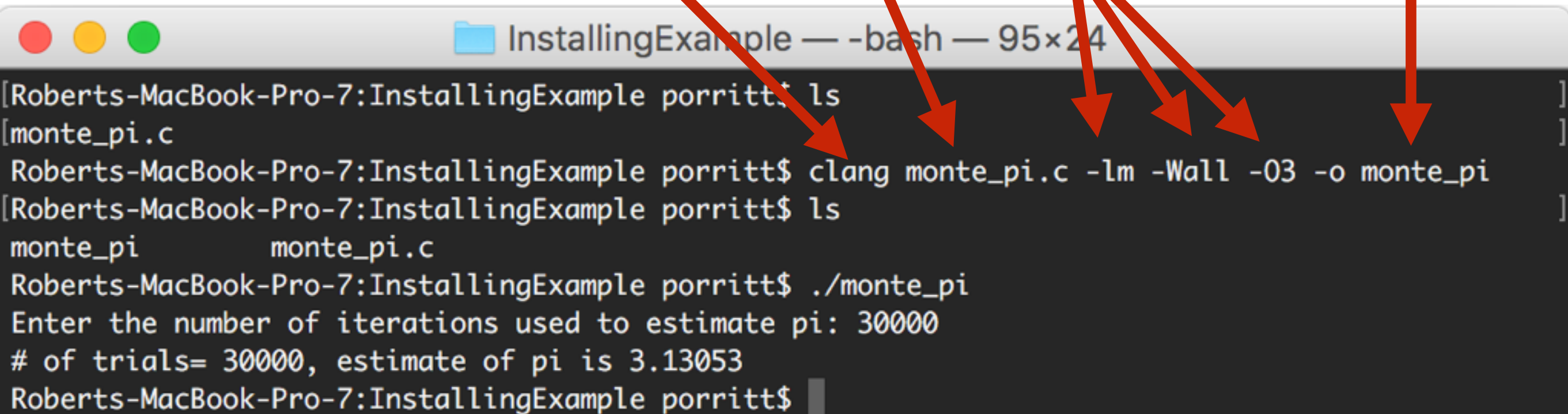
```
1  /* Program to compute Pi using Monte Carlo methods */
2
3  /* http://www.dartmouth.edu/~rc/classes/soft\_dev/C\_simple\_ex.html */
4
5  #include <stdlib.h>
6  #include <stdio.h>
7  #include <math.h>
8  #include <string.h>
9  #include <time.h>
10 #define SEED 35791246
11
12 int main(int argc, char *argv[])
13 {
14     int niter=0;
15     double x,y;
16     int i,count=0; /* # of points in the 1st quadrant of unit circle */
17     double z;
18     double pi;
19     time_t t;
20
21     printf("Enter the number of iterations used to estimate pi: ");
22     scanf("%d",&niter);
23
24     /* initialize random numbers */
25     // srand(SEED);
26     srand((unsigned) time(&t));
27     count=0;
28     for ( i=0; i<niter; i++) {
29         x = (double)rand()/RAND_MAX;
30         y = (double)rand()/RAND_MAX;
31         z = x*x+y*y;
32         if (z<=1) count++;
33     }
34     pi=(double)count/niter*4;
35     printf("# of trials= %d, estimate of pi is %g \n",niter,pi);
36     return 0;
37 }
38
39
```

c compiler

input

flags

output



```
InstallingExample — -bash — 95x24
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
monte_pi.c
Roberts-MacBook-Pro-7:InstallingExample porritt$ clang monte_pi.c -lm -Wall -O3 -o monte_pi
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
monte_pi      monte_pi.c
Roberts-MacBook-Pro-7:InstallingExample porritt$ ./monte_pi
Enter the number of iterations used to estimate pi: 30000
# of trials= 30000, estimate of pi is 3.13053
Roberts-MacBook-Pro-7:InstallingExample porritt$
```

The image shows a terminal window with a dark background and light text. The window title is "InstallingExample — -bash — 95x24". The terminal content shows a series of commands and their outputs. Red arrows point from labels at the top to specific parts of the terminal text: "c compiler" points to "clang", "input" points to "monte_pi.c", "flags" points to "-lm -Wall -O3", and "output" points to "monte_pi".

```
1  /* Edit of monte_pi to use a subroutine */
2
3  #include <stdlib.h>
4  #include <stdio.h>
5  #include <math.h>
6  #include <string.h>
7  #include <time.h>
8
9  int count_it(int count);
10
11 int main(int argc, char *argv[])
12 {
13     int niter=0;
14     int i,count=0; /* # of points in the 1st quadrant of unit circle */
15     double pi;
16     time_t t;
17
18     printf("Enter the number of iterations used to estimate pi: ");
19     scanf("%d",&niter);
20
21     /* initialize random numbers */
22     // srand(SEED);
23     srand((unsigned) time(&t));
24     count=0;
25     for ( i=0; i<niter; i++) {
26         count=count_it(count);
27     }
28     pi=(double)count/niter*4;
29     printf("# of trials= %d, estimate of pi is %g \n",niter,pi);
30     return 0;
31 }
32
33
```

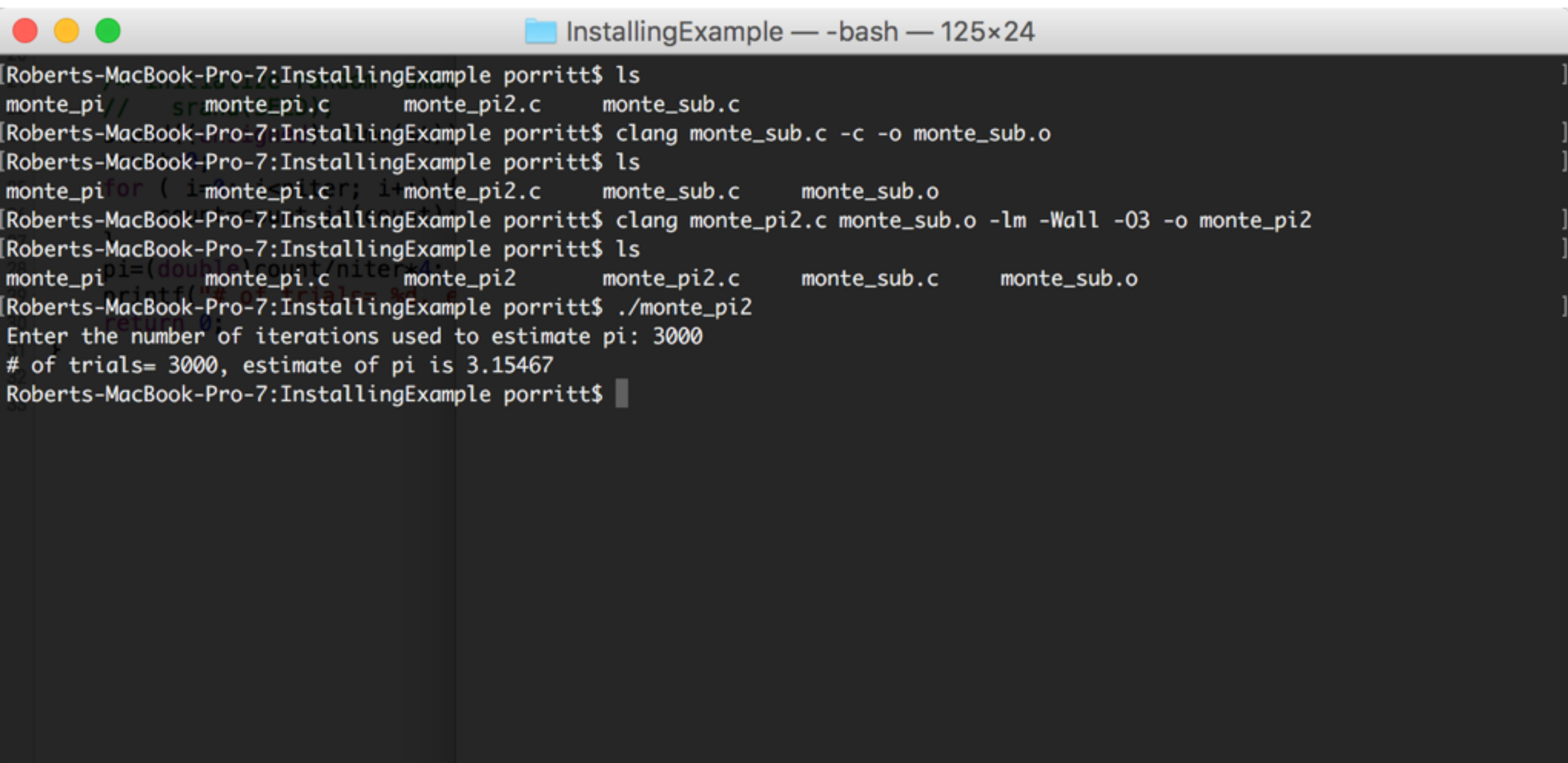
Contents of subroutine

monte_sub.c

monte_sub.c > count_it()

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <math.h>
4
5 int count_it(int count) {
6     double x, y, z;
7     x = (double)rand()/RAND_MAX;
8     y = (double)rand()/RAND_MAX;
9     z = x*x+y*y;
10    if (z<=1) count++;
11    return count;
12 }
```


Separately compile the subroutine with -c and .o extension. Then compile together



```
InstallingExample — -bash — 125x24
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
monte_pi.c  monte_pi2.c  monte_sub.c
[Roberts-MacBook-Pro-7:InstallingExample porritt$ clang monte_sub.c -c -o monte_sub.o
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
monte_pi.c  monte_pi2.c  monte_sub.c  monte_sub.o
[Roberts-MacBook-Pro-7:InstallingExample porritt$ clang monte_pi2.c monte_sub.o -lm -Wall -O3 -o monte_pi2
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
monte_pi.c  monte_pi2  monte_pi2.c  monte_sub.c  monte_sub.o
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ./monte_pi2
Enter the number of iterations used to estimate pi: 3000
# of trials= 3000, estimate of pi is 3.15467
[Roberts-MacBook-Pro-7:InstallingExample porritt$
```



```
CC = clang
FLAGS = -lm -Wall -O3
INSTALL_DIR = ~/bin/
BIN = monte_pi
SUBS = monte_sub.c
SRC = monte_pi2.c

all :: $(BIN)

monte_pi ::
    $(CC) $(SUBS) -c $(FLAGS) -o subs.o
    $(CC) $(SRC) subs.o $(FLAGS) -o $(BIN)

install ::
    install -s $(BIN) $(INSTALL_DIR)

clean ::
    rm subs.o $(BIN)
```



```
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
Makefile      monte_pi.c    monte_pi2.c   monte_sub.c
[Roberts-MacBook-Pro-7:InstallingExample porritt$ make
clang monte_sub.c -c -lm -Wall -O3 -o subs.o
clang: warning: -lm: 'linker' input unused
clang monte_pi2.c subs.o -lm -Wall -O3 -o monte_pi
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ls
Makefile      monte_pi      monte_pi.c    monte_pi2.c   monte_sub.c   subs.o
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ./monte_pi
Enter the number of iterations used to estimate pi: 3000
# of trials= 3000, estimate of pi is 3.12933
[Roberts-MacBook-Pro-7:InstallingExample porritt$ make install
install -s monte_pi ~/bin/
[Roberts-MacBook-Pro-7:InstallingExample porritt$ monte_pi
Enter the number of iterations used to estimate pi: 3000
# of trials= 3000, estimate of pi is 3.184
[Roberts-MacBook-Pro-7:InstallingExample porritt$ make clean
rm subs.o monte_pi
[Roberts-MacBook-Pro-7:InstallingExample porritt$ ./monte_pi
-bash: ./monte_pi: No such file or directory
[Roberts-MacBook-Pro-7:InstallingExample porritt$ monte_pi
Enter the number of iterations used to estimate pi: 3000
# of trials= 3000, estimate of pi is 3.15067
[Roberts-MacBook-Pro-7:InstallingExample porritt$
```

```
BIN = convolutionTests
CC = gcc
OBJ = convolutionTests.o
LIBS = -L/usr/local/lib -lfftw3 -lm
FLAGS = -Wall -O2 -I/usr/local/include
INSTALL_DIR = ~/bin

all ::
    $(CC) $(OBJ) $(LIBS) $(FLAGS) -o $(BIN)

install ::
    install -s $(BIN) $(INSTALL_DIR)

clean ::
    rm $(BIN)
```

```
~
~
~
~
~
~
~
"Makefile" 17L, 271C
```



[Download](#) [GitHub](#) [Mailing List](#)  [Benchmark](#) [Features](#) [Documentation](#) [FAQ](#) [Links](#) [Feedback](#)


Introduction

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which is [free software](#), should become the [FFT](#) library of choice for most applications.

The latest official release of FFTW is version **3.3.4**, available from [our download page](#). Version 3.3 introduced support for the AVX x86 extensions, a distributed-memory implementation on top of MPI, and a Fortran 2003 API. Version 3.3.1 introduced support for the ARM Neon extensions. See the [release notes](#) for more information.

The FFTW package was developed at [MIT](#) by [Matteo Frigo](#) and [Steven G. Johnson](#).

Our [benchmarks](#), performed on a variety of platforms, show that FFTW's performance is typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes. In contrast to vendor-tuned codes, however, FFTW's performance is *portable*: the same program will perform well on most architectures without modification. Hence the name, "FFTW," which stands for the somewhat whimsical title of "**Fastest Fourier Transform in the West**."

Subscribe to the [fftw-announce mailing list](#) to receive release announcements (or use the web feed ).

Features

FFTW 3.3.4 is the latest official version of FFTW (refer to the [release notes](#) to find out what is new). Here is a list of some of FFTW's more interesting features:



```
Roberts-MacBook-Pro-7:FFTW porritt$ ls
fftw-3.3.4.tar
Roberts-MacBook-Pro-7:FFTW porritt$ tar -xvf fftw-3.3.4.tar
x fftw-3.3.4/
x fftw-3.3.4/mpi/
x fftw-3.3.4/mpi/f03api.sh
x fftw-3.3.4/mpi/Makefile.am
x fftw-3.3.4/mpi/genf03-wrap.pl
x fftw-3.3.4/mpi/transpose-alltoall.c
x fftw-3.3.4/mpi/rdft2-rank-geq2.c
x fftw-3.3.4/mpi/mpi-dft.h
x fftw-3.3.4/mpi/fftw3-mpi.h
x fftw-3.3.4/mpi/fftw3-mpi.f03.in
x fftw-3.3.4/mpi/rdft2-problem.c
x fftw-3.3.4/mpi/dft-rank1.c
x fftw-3.3.4/mpi/block.c
x fftw-3.3.4/mpi/rdft2-serial.c
x fftw-3.3.4/mpi/rdft-problem.c
x fftw-3.3.4/mpi/mpi-rdft2.h
x fftw-3.3.4/mpi/testsched.c
x fftw-3.3.4/mpi/rdft2-solve.c
x fftw-3.3.4/mpi/any-true.c
```

```

[Roberts-MacBook-Pro-7:FFTW porritt$ ls
fftw-3.3.4  fftw-3.3.4.tar
[Roberts-MacBook-Pro-7:FFTW porritt$ cd fftw-3.3.4
[Roberts-MacBook-Pro-7:fftw-3.3.4 porritt$ ls
AUTHORS      Makefile.am  api          configure    genfft       missing      tests
CONVENTIONS  Makefile.in  bootstrap.sh configure.ac  install-sh   mpi          threads
COPYING      NEWS         compile      depcomp      kernel       rdft         tools
COPYRIGHT    README       config.guess dft          libbench2    reodft
ChangeLog    TODO         config.h.in  doc          ltmain.sh    simd-support
INSTALL      aclocal.m4   config.sub   fftw.pc.in   m4           support
Roberts-MacBook-Pro-7:fftw-3.3.4 porritt$

```

General install from source:

1. `tar -xvf software.tar.gz`
2. `cd {software directory}`
3. `./configure`
4. `make`
5. `make install`

Voila!

```

config.status: creating tests/Makefile
config.status: creating doc/Makefile
config.status: creating doc/FAQ/Makefile
config.status: creating tools/Makefile
config.status: creating tools/fftw_wisdom.1
config.status: creating tools/fftw-wisdom-to-conf
config.status: creating m4/Makefile
config.status: creating fftw.pc
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
Roberts-MacBook-Pro-7:fftw-3.3.4 porritt$

```

Some packages installed this way...

- mseed2sac
- evalresp
- sac (also pre-compiled versions are available)
- seismic unix
- Computer Programs in Seismology (slightly modified)

Package Management

How do you deal with software dependencies?

How do you deal with hardware dependencies?

How do you build new programs with old tools?



Command-line package managers



Homebrew

Mac Ports



Debian/Ubuntu: apt-get

Fedora/CentOS: yum



pip

anaconda: conda

Install

Desktop — -bash — 79×24

```
[Roberts-MacBook-Pro-7:Desktop porritt$ brew install emojiify  
=> Downloading https://github.com/mrowa44/emojiify/archive/v1.0.2.tar.gz  
=> Downloading from https://codeload.github.com/mrowa44/emojiify/tar.gz/v1.0.2  
##### 100.0%  
🍺 /usr/local/Cellar/emojiify/1.0.2: 4 files, 43.5K, built in 1 second  
Roberts-MacBook-Pro-7:Desktop porritt$
```

Run

Desktop — -bash — 59×19

```
Roberts-MacBook-Pro-7:Desktop porritt$ emojiify "To :bee: ,  
or not to :bee: : that is the question... To take :muscle:  
against a :ocean: of troubles, and by opposing, end them?"  
To 🐝, or not to 🐝: that is the question... To take 💪agai  
nst a 🌊 of troubles, and by opposing, end them? Roberts-Mac  
Book-Pro-7:Desktop porritt$
```

Example, installing GMT5

Red Hat, CentOS, Fedora:

```
$ sudo yum install GMT gshhg-gmt-nc4-all dcw-gmt
```

Mac OSX

Download installer and add to path

or

```
$ sudo port install gdal +curl +geos +hdf5 +netcdf
```

```
$ sudo port install gmt5
```

or

```
$ sudo fink install gmt5
```

or

```
$ brew install homebrew/science/gmt
```

Windows

Download and run installer

Seems uncomfortable to use package managers?

You already use them everyday...



Some packages installed this way...

- obspy (also installs numpy, matplotlib, etc...)
- gmt
- git

Installing scripts or pre-compiled software (SOD, TauP, Fetch scripts, etc...)

The screenshot shows a Mac desktop environment. At the top, the menu bar includes 'QuickTime Player', 'File', 'Edit', 'View', 'Window', and 'Help'. The system status bar on the right shows 'Tue 1:43 PM' and '100%' battery. The desktop background is a dark, abstract image. A web browser window is open, displaying the 'USC Seismology - TauP' website. The browser's address bar shows 'www.seis.sc.edu/taup/'. The website has a navigation menu with links: 'Home', 'People', 'Projects', 'Software' (highlighted), 'SCSN', 'K-12', and 'Contact'. On the left side of the website is a logo for 'Lithospheric Seismology' with the text 'AT THE UNIVERSITY OF SOUTH CAROLINA'. The main content area of the website is titled 'TauP' and contains the following text:

You are here: [Software](#) > [TauP](#)

TauP

The TauP Toolkit is a seismic travel time calculator. In addition to travel times, it can calculate derivative information such as ray paths through the earth, pierce and turning points. It handles many types of velocity models and can calculate times for virtually any seismic phase with a phase parser. It is written in Java so it should run on any Java enabled platform. Unfortunately, glibc, which comes with many Linux distributions, is not quite Java and prevents TauP from finding its model files. Please use Java from Sun on Linux instead.

Download

We have released the next version of TauP, [TauP-2.4.0](#), which fixes an issue with the output of `TauP_Path` for phases like `Pn`, documentation fixes and improvements when using TauP as a library.

You may download the latest version, 2.4.0, as a [tar.gz](#) under the terms of the [Gnu Public License](#). It was created on July 6, 2016.

Documentation

The documentation for TauP is available as a [PDF](#) document. You can download the [zip file](#) or the [tar.gz](#) version containing a draft and the source images of our paper submitted to the Seismological Research Letters.

If you would like TauP and would like to cite it, please use the following: Crotwell, H. P., T. J. Owens, and J. Ritsema (1999). The TauP Toolkit: Flexible seismic travel-time and ray-path utilities, *Seismological Research Letters* 70, 154-160.

If you have any comments, please [email us](#). You may also join the [TauP email list](#).

The TauP Toolkit source code is hosted on github here: <https://github.com/crotwell/TauP>.

The TauP Toolkit jar is also available from [maven central](#) with group `edu.sc.seis` and id `TauP`.

At the bottom of the website, the footer reads: 'University of South Carolina, Department of Earth and Ocean Sciences, 701 Sumter Street Columbia, SC 29208 • [Email](#)'.

In the bottom right corner of the desktop, a terminal window is open, showing a prompt 'bash' and a window size of '160x41'.

Installing scripts or pre-compiled software (SOD, TauP, Fetch scripts, etc...)

```
[guest-wireless-upc-1606-10-120-106-164:~ porritt$ ls -lh FetchData
```

```
-rw-r-----@ 1 porritt  staff    72K Jul 14 12:36 FetchData
```

```
[guest-wireless-upc-1606-10-120-106-164:~ porritt$ head FetchData
```

```
#!/usr/bin/perl
```

Tells shell what program to run this under

```
#
```

```
# FetchData
```

```
#
```

```
# Find the most current version at http://service.iris.edu/clients/
```

```
#
```

```
# Fetch data and related metadata from web services. The default web
```

```
# service are from the IRIS DMC, other FDSN web services may be
```

```
# specified by setting the following environment variables:
```

```
#
```

Adds permission to execute

```
[guest-wireless-upc-1606-10-120-106-164:~ porritt$ ./FetchData
```

```
-bash: ./FetchData: Permission denied
```

```
[guest-wireless-upc-1606-10-120-106-164:~ porritt$ chmod +x FetchData
```

```
[guest-wireless-upc-1606-10-120-106-164:~ porritt$ ./FetchData
```

```
FetchData: collect time series and related metadata (version 2015.246)
```

```
http://service.iris.edu/clients/
```

```
Usage: FetchData [options]
```

```
Options:
```

```
-v          Increase verbosity, may be specified multiple times
```

```
-q          Be quiet, do not print anything but errors
```

```
-N,--net    Network code, list and wildcards (* and ?) accepted
```

```
-S,--sta    Station code, list and wildcards (* and ?) accepted
```

Installing programs for Matlab

Check the file exchange:

<https://www.mathworks.com/matlabcentral/fileexchange/>

Hope the authors include an install or setup script

Add executable scripts to matlab path (ie pathtool or addpath)

Add jar files (ie group of java software files) to either the dynamic java path (javaaddpath) or the static class path
edit ~/.matlab/{Version}/classpath.txt

Thanks! Any questions?