

star

[About](#) | [News](#) | [Research](#) | [Teaching](#) | [Publications](#) | [Outreach](#) | [People](#) | [Download](#) | [Tools](#) | [Links](#) | [Contact](#) |  [RSS 2.0](#)**You are here:** [Department](#) / [Mad Star Home](#) / [Download Stuff](#) / [MESA SDK](#)

MESA SDK

The MESA SDK is a collection of compilers and run-time libraries that make it easy to install and use the [MESA](#) stellar evolution code.

1. [Overview](#)
2. [Linux](#)
 1. [Compatibility](#)
 2. [Prerequisites](#)
 3. [Download](#)
 4. [Installation](#)
 5. [Usage](#)
3. [Mac OS X](#)
 1. [Compatibility](#)
 2. [Prerequisites](#)
 3. [Download](#)
 4. [Installation](#)
 5. [Usage](#)
4. [Making Movies](#)
5. [Installing as Root User](#)
6. [Troubleshooting](#)
7. [Frequently Asked Questions \(FAQ\)](#)

Overview

Experience has demonstrated that incompatibilities and bugs in compilers and libraries are significant obstacles in getting MESA up and running with minimal fuss. For instance, MESA makes use of various features in the new(ish) Fortran 2003 standard, which aren't implemented (or are still buggy) in all but the most recent compiler releases.

To help overcome these obstacles, I've put together a unified software development kit (SDK) which contains compilers and libraries *known* to compile MESA correctly. The SDK contains the following components:

- The [GNU Comiplier Collection \(gcc\)](#), with support for C, C++, and Fortran
- The [GNU Project Debugger \(gdb\)](#)
- The [Valgrind](#) code instrumentation framework
- The [Basic Linear Algebra Subprogram \(BLAS\)](#) library
- The [Linear Algebra PACKage \(LAPACK\)](#) library
- The [Hierarchical Data Format v5 \(HDF5\)](#) library
- The [PGPLOT](#) graphics library
- The [SE](#) library from the NuGrid project
- The [ffmpeg](#) movie encoder
- The [ndiff](#) fuzzy comparison tool
- Various helper scripts for use in linking against these libraries and other tasks

Currently, both [Linux](#) and [Mac OS X](#) running on Intel/AMD processors are supported. Although the SDK was initially bundled as part of the standard MESA distribution (from release 3708 onwards), it makes more sense to keep it separate. This page hosts all the necessary information and links to download, install and use the SDK.

Linux

Compatibility

The SDK should work on any relatively-recent Linux distribution running on 64-bit Intel-compatible processors (32-bit processors are no longer supported; in any case, MESA itself doesn't work on 32-bit). The GNU C library (also known as

GLIBC) included in the distribution must be version 2.5 or more recent; to determine what GLIBC your system uses, run the command `/lib/libc.so.6` (or, possibly, `/lib64/libc.so.6`) and examine the first line of the output for the version number.

Prerequisites

The following components must be installed for the SDK to work on Linux-based systems:

- The 'Binutils' development tools
- The 'Make' dependency/compilation tool
- The 'Perl' scripting language
- The 'X11' windowing library plus development headers
- The 'Z' compression library plus development headers
- The 'C' shell or derivatives

Not all of these components are installed by default on some Linux distributions, and you may have to use the appropriate package management tool (e.g., apt-get, yum, rpm, emerge) to install them. The following table lists the package names of the components (and any other pieces that are required) for some of the more-common distributions:

Package	Fedora / CentOS	Ubuntu	Mint	Gentoo	Arch
Binutils	binutils	binutils	binutils	sys-devel/binutils	binutils
Make	make	make	make	sys-devel/make	make
Perl	perl	perl	perl	dev-lang/perl	perl
X11 library	libX11, libX11-devel	libx11-6, libx11-dev	libx11-dev	x11-libs/libX11	libx11
Z library	zlib, zlib-devel	zlib1g, zlib1g-dev	zlib-dev	sys-libs/zlib	zlib
C shell	tcsh	tcsh	tcsh	sys-shells/tcsh	tcsh
Other			libc6-dev		glibc

If your distribution is not listed here, please [contact](#) me and I'll add it to the table.

Download

To download the SDK for Linux, click on the appropriate link in the table:

Release Date MESA Version at Release	File	Notes
March 15 2019 (current) 11554	mesasdk-x86_64-linux-20190315.tar.gz	Updated to gcc 8.3.0; many other packages also updated
August 22 2018 10398	mesasdk-x86_64-linux-20180822.tar.gz	Fixed 'terminals database is inaccessible' (and similar) errors, by moving ncurses binaries out of the path

Note that versions of the SDK older than the current one are not formally supported but are provided [here](#) as a courtesy; if you run into problems using an older version, you should first try upgrading to the current version.

Installation

On Linux the SDK can be installed anywhere (this is different from previous releases, where it had to be put in the `/opt` directory). However, for simplicity the following instructions will assume you're installing in your home directory. The steps are as follows:

- Download the package from the [table above](#)
- Extract it using the command `tar xvfz package_name -C ~/` (note that's a tilde in front of the slash!)
- Set the path to the SDK:
 - For the C shell: `setenv MESASDK_ROOT ~/mesasdk`
 - For the Bourne shell: `export MESASDK_ROOT=~/mesasdk`
- Initialize the SDK (also checks compatibility):
 - For the C shell: `source $MESASDK_ROOT/bin/mesasdk_init.csh`
 - For the Bourne shell: `source $MESASDK_ROOT/bin/mesasdk_init.sh`
- Check that the SDK is properly installed by running `gfortran --version`. The first line of the output should look something like this:
GNU Fortran (GCC) 7.3.0
If it *doesn't*, then you should consult the [Troubleshooting](#) section below.

Steps 3 and 4 need to be repeated each time you begin a new shell session; alternatively, they can be added to the appropriate shell start-up file (`~/.cshrc` for the C shell, and `~/.profile` for the Bourne shell).

Usage

Nothing special needs to be done in order to use the SDK to build MESA. Simply change into the top-level `mesa` directory (the one created when you download MESA via `svn`) and then run `./install`.

Mac OS X

Compatibility

The SDK should work on any relatively-recent OS X distribution (10.10, Yosemite or later) running on Apple computers with 64-bit Intel-compatible processors (32-bit processors are no longer supported; in any case, MESA itself doesn't work on 32-bit).

Prerequisites

The following components must be installed for the SDK to work on OS X systems:

- The Xcode command-line tools
- The XQuartz X-windows infrastructure

The Xcode command-line tools can be installed by running `xcode-select --install` from a Terminal prompt. (Note that a full installation of the Xcode development environment is not necessary). Likewise, XQuartz can be downloaded and installed from [here](#). Note that it is often necessary to *reinstall* the command-line tools and/or XQuartz after upgrading to a new release of OS X.

IMPORTANT NOTE: On OS X 10.14 (Mojave), you may also need to install the development header files. These ship as a standard part of Mojave, but must be installed by hand. To do so, run the command

```
open /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg
```

Download

To download the SDK for OS X, click on the appropriate link in the table:

Release Date MESA Version at Release	File OS X 10.12, Sierra OS X 10.13, High Sierra	File OS X 10.14, Mojave	Notes
March 15 2019 (current) 11554	mesasdk-x86_64-osx-10.12-10.14-20190315.dmg		Updated to gcc 8.3.0; many other packages also updated
November 4 2018 10398		mesasdk-x86_64-osx-10.14-20181104.dmg	Special release for OS X 10.14 (Mojave)

Note that versions of the SDK older than the current one are not formally supported but are provided [here](#) as a courtesy; if you run into problems using an older version, you should first try upgrading to the current version.

Installation

On OS X the SDK must be installed in the Applications folder (although since the October 7 2018 release it can be installed elsewhere, if desired). The steps are as follows:

- Download the disk image from the [table above](#)
- Open it by double clicking on it in the Finder
- Drag the `mesasdk` folder across to the Applications folder
- Set the path to the SDK:
 - For the C shell: `setenv MESASDK_ROOT /Applications/mesasdk`
 - For the Bourne shell: `export MESASDK_ROOT=/Applications/mesasdk`
- Initialize the SDK (also checks compatibility):
 - For the C shell: `source $MESASDK_ROOT/bin/mesasdk_init.csh`
 - For the Bourne shell: `source $MESASDK_ROOT/bin/mesasdk_init.sh`
- Check that the SDK is properly installed by running `gfortran --version`. The first line of the output should look something like this:
GNU Fortran (GCC) 7.3.0
If it *doesn't*, then you should consult the [Troubleshooting](#) section below.

Steps 4 and 5 need to be repeated each time you begin a new shell session; alternatively, they can be added to the appropriate shell start-up file (`~/.cshrc` for the C shell, and `~/.profile` for the Bourne shell).

Usage

See the [usage instructions](#) above for Linux (they are the same for OS X).

Making Movies

The SDK includes the `ffmpeg` encoder and a simple script, `images_to_movie.sh`, which uses `ffmpeg` to create movies from PNG files produced by MESA.

To illustrate the script in action, suppose the `&pgstar` section of the MESA inlist file contains the following parameters:

```
:
Grid6_file_flag = .true.
Grid6_file_dir = 'png'
Grid6_file_prefix = 'grid6_'
:
```

This will make MESA write a sequence of PNG images into the `png` subdirectory, with filenames `grid6_NNNNN.png` (where `N` represents a single digit). To combine these files together into a movie, run the following command from the same directory MESA was run in:

```
images_to_movie.sh 'png/grid6_*.png' movie.mp4
```

This will produce an MPEG4 movie, with the filename `movie.mp4`. The type of movie produced is determined from the file extension. Other choices are possible (e.g., `mpg` for MPEG2), but the `images_to_movie.sh` script is specifically targeted at producing MPEG4 output, so it might be best to stick with the `mp4` extension unless you know what you're doing.

IMPORTANT NOTE: In the example above, the quotes (' ') are necessary to prevent the shell from prematurely expanding the wildcard (*) character.

Installing as Root User

If you wish to install the SDK as the root user, then that's fine --- but you must run the initialization script at least once as root, before you can use the SDK as an ordinary user. This is because configuration files are written into the `$MESASDK_ROOT/etc` directory on first initialization; and these files must be written as root if you installed as root.

Troubleshooting

If you encounter an error during the build process, please consult the [FAQ](#) first. If your problem is not resolved, then you should submit a bug report to the [MESA Forum](#). Be sure to include the following information in your bug report:

- The version of MESA you are trying to build
- The version of the SDK you are using (use the `mesasdk_version.sh` command to determine this)
- Your platform (machine type, operating system, version)

Also, it would be helpful if you could post the output of the following commands:

- `uname -a`
- `gfortran -v`
- `echo $MESASDK_ROOT`
- `echo $PATH`

Frequently Asked Questions (FAQ)

Q: When trying to download the SDK using `wget`, I get the error

```
403: Forbidden
```

A: Our web server is set up to reject requests from `wget`. As a workaround, add the flag `--user-agent=""` to your `wget` invocation.

Q: How can I download the SDK from the command line?

A: Use the `wget` tool. For instance, to download the Linux version dated YYYYMMDD, run

```
wget --user-agent="" http://www.astro.wisc.edu/~townsend/resource/download/mesasdk/mesasdk-x86_64-linux-
YYYYMMDD.tar.gz
```

(See the question above for a discussion of why the `--user-agent=""` flag is necessary.)

Q: I'm getting compilation errors of the form:

```
/usr/bin/ld: cannot find -lX11
/usr/bin/ld: cannot find -lz
```

A: Have you properly installed the X windows and Z compression libraries, as specified in the [prerequisites](#)?

Q: I'm getting compilation errors of the form:

```
libpng warning: Application built with libpng-1.2.10 but running with 1.5.6
PGPLOT /png: error in libpng while writing file...
```

A: This is a known problem, caused by the `pgplot` library being compiled with the wrong `libpng` headers. It was fixed in the 20120727 release of the SDK; if you are using an older release, please upgrade.

Q: I'm getting compilation errors of the form:

```
../private/utils_isnan_okay.f:43.36:
      is_real_inf = (2*x==x .and. x /= 0)
                   1
Warning: Inequality comparison for REAL(4) at (1)
```

A: This is a known issue with the 20130320 (and later) releases of the SDK, caused by the upgrade to `gfortran` 4.8.0. To fix, add the flag `'-Wno-compare-reals'` to the end of the definition of the `FCwarn` variable in `mesa/utils/makefile_header`.

Q: I'm getting compilation errors of the form:

```
../private/utils_dict.f:429.41:
      integer, parameter :: multiplier = 31
                   1
Warning: Unused parameter 'multiplier' declared at (1)
```

A: This is a known issue with the 20130320 (and later) releases of the SDK, caused by the upgrade to `gfortran` 4.8.0. To fix, add the flag `'-Wno-unused-parameter'` to the end of the definition of the `FCwarn` variable in `mesa/utils/makefile_header`.

Q: On Ubuntu Linux I encounter these errors during compilation:

```
/usr/bin/ld: cannot find crt1.o: No such file or directory
/usr/bin/ld: cannot find crti.o: No such file or directory
collect2: error: ld returned 1 exit status
```

A: This is a known problem, caused by Ubuntu's use of non-standard installation locations. It was fixed in the 20120727 release of the SDK; if you are using an older release, please upgrade.

Q: On Ubuntu Linux I encounter this error during compilation:

```
/opt/mesasdk/lib/gcc/i686-pc-linux-gnu/4.7.0/include-fixed/features.h:338:25: fatal error: sys/cdefs.h: No such
file or directory compilation terminated.
make: *** [btf_order.o] Error 1
```

A: This is a known problem, caused by Ubuntu's use of non-standard installation locations. It was fixed in the 20120727 release of the SDK; if you are using an older release, please upgrade.

Q: On Red Hat Enterprise Linux (RHEL) I encounter this error during compilation:

```
gfortran: /lib/libc.so.6: version `GLIBC_2.11' not found (required by gfortran)
```

A: This is a known problem, caused by the SDK being compiled with a more-recent version of the GNU C Library (GLIBC) than is installed on RHEL systems. It was fixed in the 20120120 release of the SDK; if you are using an older release, please upgrade.

Q: On OS X I encounter this error during compilation:

```
Re: dyld: unknown required load command 0x80000022
```

A: This problem likely stems from trying to use the SDK on an older version of OS X (10.4 Tiger or 10.5 Leopard), as these have difficulty running 64-bit executables. Please contact Rich Townsend for further support.

Q: On OS X 10.8 I find an error such as

```
dyld: lazy symbol binding failed: Symbol not found: ___emutls_get_address
Referenced from: /usr/local/lib/libgomp.1.dylib
Expected in: /usr/lib/libSystem.B.dylib

This is a run-time error. It shows up, for example, in the output of a module test such as const/test/tmp.txt.
```

A: Try adding both the \$MESASDK_ROOT/lib and /usr/local/lib directories to your DYLD_LIBRARY_PATH environment variable. These should appear first and second, respectively, in the path.

Q: I tried to compile MESA on OS X 10.9, and I got the following error

```
/usr/bin/awk: can't open file ../ndiff/share/lib/ndiff/ndiff-2.00/ndiff.awk
source line number 1 source file ../ndiff/share/lib/ndiff/ndiff-2.00/ndiff.awk
```

A: This means ndiff didn't compile correctly, most likely because the command-line tools weren't installed. See the note in the [prerequisites](#).

Q: On Linux systems gcc can't create executables, producing errors of the sort

```
In file included from test.c:1:0:
/usr/include/stdio.h:320:43: error: missing binary operator before token "("
  #if defined __USE_XOPEN2K8 || __GLIBC_USE (LIB_EXT2)
```

A: This is occurring because the C header files that ship with the SDK need to be updated to work properly with the system header files. To update the header files, run the following commands after setting MESASDK_ROOT:

```
GCC_VERSION=`gcc --version | grep ^gcc | sed 's/^.* //g'`
$MESASDK_ROOT/libexec/gcc/x86_64-pc-linux-gnu/$GCC_VERSION/install-tools/mkheaders $MESASDK_ROOT
```

Q: The initialization script produces errors of the sort

```
mesasdk_init.sh: checking architecture
touch: cannot touch '/opt/mesasdk/etc/check_arch.done': Permission denied
```

A: This is occurring because you installed the SDK as a different user (e.g., root). You must run the initialization script at least once as the installation user, before running as any other user.

Q: On OS X 10.14 (Mojave) I encounter this error:

```
configure: error: C compiler cannot create executables
```

See `config.log` for more details.

A: This may be occurring because you haven't installed the development header files. Check that the directory `/usr/include` exists; if it is missing, then follow the instructions given in the OS X Prerequisites section, above, to install the header files.

Q: When compiling, I'm encountering the error

```
file locking disabled on this file system (use HDF5_USE_FILE_LOCKING environment variable to override)
```

A: To fix this, set the `HDF5_USE_FILE_LOCKING` environment variable to `FALSE`. This can be done via the command

```
export HDF5_USE_FILE_LOCKING=FALSE
```

Updated 2019-03-19 14:43:54