



Compilers and Debuggers

Nikolaos Nikoloutsakos

Greek Research & Technology Network

Athens, 22 Nov. 2018



Contents

- 1 Environment Modules
 - Available compilers
 - MPI compilers
- 2 Compilers
 - Compilation flags
 - Optimization flags
- 3 Debuggers
 - GDB
 - DDD
 - Parallel Debugging
 - Debugging OpenMP Applications
 - Debugging MPI Applications
 - Valgrind



Environment

- Login Shell

```
/bin/bash
```

```
echo $SHELL
```

- Environment Variables

```
env
```

```
printenv GRNET_ROOT; echo $GRNET_ROOT
```

- Default variables
 - ▶ \$GRNET_ROOT
 - ▶ \$HOME
 - ▶ \$WORKDIR



Environment Modules

- **environment modules** to manage user environment
- Access applications or versions on ARIS
- Dynamically set up environments for different applications **PATH**, **LD_LIBRARY_PATH** etc.
- One module for each software version
- Detects software dependencies and informs which modules must be loaded



What modules exist?

module avail

List available modules.

```
----- /apps/modulefiles/applications -----  
abinit/7.10.4(default)          namd/2.10/hybrid/normal  
bigdft/1.7.6(default)          namd/2.10/purempi/memopt  
cdo/1.7.0(default)             namd/2.10/purempi/normal  
code_saturne/4.0.1/intel       ncarg/6.3.0(default)
```

What modules exist?

```
module avail <module>
```

```
List <module> available versions
```

```
/apps/modulefiles/compilers:  
gnu/4.9.2(default)  
gnu/4.9.3  
gnu/5.1.0  
gnu/5.2.0
```

Default version marked as **(default)**



● Default module version

- ▶ Almost all ARIS software packages have multiple versions marked as **(default)** In these case the commands:

```
module load MODULENAME
```

and

```
module load MODULENAME/DEFAULTVERSION
```

are equivalent.

- ▶ eg. current defaults

```
module load intel
```

and

```
module load intel/15.0.3
```

are the same



- ▶ Sometimes, after proper notification, the system defaults may change.
- ▶ In that case if you are using the `defaults`, it's recommended to re-compile you code.
- ▶ If you need to use a specific software version, please use it's relevant module version.



Use Modules

`module load/unload <module>`

Load/Unload <module> into the shell environment

`module switch <module1> <module2>`

Switch loaded module1 with module2.

`module list`

List loaded modules.

```
Currently Loaded Modulefiles:
```

```
1) gnu/4.9.2          2) intel/15.0.3      3) intelmpi/5.0.3
```



Examine a module

`module show <module>`

List all of the environment changes the <module> will make if loaded

```
-----  
/apps/modulefiles/compilers/gnu/4.9.2:  
  
module-whatis  Enable usage for the GNU Compiler Collection ver  
setenv        COMPILER_GNUROOT /apps/compilers/gnu/4.9.2  
prepend-path  PATH /apps/compilers/gnu/4.9.2/bin  
prepend-path  INCLUDE /apps/compilers/gnu/4.9.2/include  
prepend-path  LD_LIBRARY_PATH /apps/compilers/gnu/4.9.2/lib  
prepend-path  LD_LIBRARY_PATH /apps/compilers/gnu/4.9.2/lib64  
prepend-path  MANPATH /apps/compilers/gnu/4.9.2/share/man  
-----
```



Examine a module

- `setenv`: Set environment variable
- `prepend-path`: Prepend value to environment variable.
- `MODULENAMEROOT/include`, `MODULENAMEROOT/lib`

`module whatis <module>`

Display what is the module information

`module help <module>`

More specific help about <module>



Available Software

<http://doc.aris.grnet.gr/software/>

Software Environment / Available Modules Search

ABSIS DOCUMENTATION
Home
System Information -
Logs and Data Transfer
User Environment
Running Jobs -
SLURM - Job Script Template
Development Environment -
Software Environment -
Available Modules
Life Sciences
Computational Chemistry & Material Science
Weather Forecasting
Engineering
Numerical Libraries
Generic Data I/O Libraries
Visualisation Software
High-level Languages
Remote Visualization
FAQ
System Messages
Contact

Software Environment

In this section we present an overview of installed software.

Software package	version	jobs	GPU	Prod
awscli	2.4.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
atop	7.10.4, 7.10.0, 6.0.7, 6.0.8, 6.4.3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
atopx	20150817	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
axel	6.3.1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
axlib	3.10.2, 3.10.3, 3.11.34, 3.11.38	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
awkbook	4.2.6	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
log4j	1.7.4, 1.7.7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
binutils	2.26, 2.26, 2.27, 2.28, 2.29	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
boost	1.58.0, 1.62.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
boost-py2.7	1.58.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Development Tools

- *Compilers:* GCC, INTEL, PGI
- *Parallel:* Intel MPI, OpenMPI
- *Debug:* gdb, gdb-ia , pgdbg, ddd
- *Profilers:* VTune, Scalasca, mpiP, gprof, pgprof

Table: Compilers

GNU	C	gcc
	C++	g++
	Fortran	gfortran
Intel	C	icc
	C++	icpc
	Fortran	ifort
PGI	C	pgcc
	C++	pgc++
	Fortran	pgfortran



Compilers

module avail

All available compilers check for "compilers" section

```
----- /apps/modulefiles/compilers -----  
binutils/2.25          gnu/5.1.0          intel/17.0.4  
binutils/2.26          gnu/5.2.0          intel/17.0.5  
binutils/2.27          gnu/5.3.0          intel/18.0.0  
binutils/2.28          gnu/5.4.0          intel/18.0.1  
binutils/2.29(default) gnu/5.5.0          java/1.7.0  
clang/5.0.0(default)  gnu/6.1.0          java/1.8.0(default)  
cuda/6.5.14           gnu/6.2.0          pgi/15.5  
cuda/7.0.28           gnu/6.3.0          pgi/16.10  
cuda/7.5.18           gnu/6.4.0          pgi/16.4  
cuda/8.0.27           gnu/7.1.0          pgi/16.5  
cuda/8.0.44           gnu/7.2.0          pgi/16.7  
cuda/8.0.61(default) intel/15.0.3(default) pgi/16.9  
cuda/9.1.85           intel/15.0.6        pgi/17.1  
gdb/7.11.1            intel/16.0.0        pgi/17.10(default)
```



MPI versions

module avail

Check for "parallel" section

```
----- /apps/modulefiles/parallel -----  
intelmpi/2017.0      openmpi/1.10.0/gnu      openmpi/2.0.0/intel  
intelmpi/2017.1      openmpi/1.10.0/intel    openmpi/2.0.1/gnu  
intelmpi/2017.2      openmpi/1.10.1/gnu      openmpi/2.0.1/intel  
intelmpi/2017.3      openmpi/1.10.1/intel    openmpi/2.0.2/gnu  
intelmpi/2017.4      openmpi/1.10.2/gnu      openmpi/2.0.2/intel  
intelmpi/2017.5      openmpi/1.10.2/intel    openmpi/2.0.3/gnu  
intelmpi/2018.0      openmpi/1.10.3/gnu      openmpi/2.0.3/intel  
intelmpi/2018.1      openmpi/1.10.3/intel    openmpi/2.1.0/gnu  
intelmpi/5.0.3(default) openmpi/1.10.4/gnu      openmpi/2.1.0/intel  
intelmpi/5.1.1        openmpi/1.10.4/intel    openmpi/2.1.1/gnu  
intelmpi/5.1.2        openmpi/1.10.5/gnu      openmpi/2.1.1/intel  
intelmpi/5.1.3        openmpi/1.10.5/intel    openmpi/2.1.2/gnu  
intelmpi/5.1.3.258    openmpi/1.10.7/gnu      openmpi/2.1.2/intel
```




Intel MPI

Language	wrapper
C	mpiicc
C++	mpiicpc
Fortran	mpifort

Launcher

`srun`



module avail intelmpi

```
----- /apps/modulefiles/parallel -----  
intelmpi/2017.0          intelmpi/2017.5          intelmpi/5.1.2  
intelmpi/2017.1          intelmpi/2018.0          intelmpi/5.1.3  
intelmpi/2017.2          intelmpi/2018.1          intelmpi/5.1.3.258  
intelmpi/2017.3          intelmpi/5.0.3(default)  
intelmpi/2017.4          intelmpi/5.1.1
```

module list

```
Currently Loaded Modulefiles:  
  1) gnu/4.9.2          2) intel/15.0.3          3) intelmpi/5.0.3
```



(gnu) mpicc -show

```
gcc -I/apps/compilers/intel/impi/5.0.3.048/intel64/include -L/apps/compilers/intel/impi/5.0.3.048/intel64/lib/release_mt -L/apps/compilers/intel/impi/5.0.3.048/intel64/lib -Xlinker --enable-new-dtags -Xlinker -rpath -Xlinker /apps/compilers/intel/impi/5.0.3.048/intel64/lib/release_mt -Xlinker -rpath -Xlinker /apps/compilers/intel/impi/5.0.3.048/intel64/lib -Xlinker -rpath -Xlinker /opt/intel/mpi-rt/5.0/intel64/lib/release_mt -Xlinker -rpath -Xlinker /opt/intel/mpi-rt/5.0/intel64/lib -lmpifort -lmpi -lmpigi -ldl -lrt -lpthread
```

(intel) mpiicc -show

```
icc -I/apps/compilers/intel/impi/5.0.3.048/intel64/include -L/apps/compilers/intel/impi/5.0.3.048/intel64/lib/release_mt -L/apps/compilers/intel/impi/5.0.3.048/intel64/lib -Xlinker --enable-new-dtags -Xlinker -rpath -Xlinker /apps/compilers/intel/impi/5.0.3.048/intel64/lib/release_mt -Xlinker -rpath -Xlinker /apps/compilers/intel/impi/5.0.3.048/intel64/lib -Xlinker -rpath -Xlinker /opt/intel/mpi-rt/5.0/intel64/lib/release_mt -Xlinker -rpath -Xlinker /opt/intel/mpi-rt/5.0/intel64/lib -lmpifort -lmpi -lmpigi -ldl -lrt -lpthread
```



OpenMPI

Language	wrapper
C	mpicc
C++	mpicxx
Fortran	mpifort, mpif90

Launcher

`srun`



module avail openmpi

```
----- /apps/modulefiles/parallel --  
openmpi/1.10.0/gnu(default) openmpi/1.8.5/intel  
openmpi/1.10.0/intel       openmpi/1.8.7/gnu  
openmpi/1.8.5/gnu         openmpi/1.8.7/intel
```



```
module load openmpi/1.10.1/intel
```

```
Currently Loaded Modulefiles:
```

- | | |
|-----------------|-------------------------|
| 1) gnu/4.9.2 | 3) intelmpi/5.0.3 |
| 2) intel/15.0.3 | 4) openmpi/1.10.1/intel |

```
mpicc -show
```

```
icc -I/apps/parallel/openmpi/1.10.1/intel/include -pthread -Wl,-rpath -Wl,/apps/parallel/openmpi/1.10.1/intel/lib -Wl,--enable-new-dtags -L/apps/parallel/openmpi/1.10.1/intel/lib -lmpi
```



```
module load openmpi/1.10.1/gnu
```

```
Currently Loaded Modulefiles:
```

- | | |
|-----------------|-----------------------|
| 1) gnu/4.9.2 | 3) intelmpi/5.0.3 |
| 2) intel/15.0.3 | 4) openmpi/1.10.1/gnu |

```
mpicc -show
```

```
gcc -I/apps/parallel/openmpi/1.10.1/gnu/include -pthread -Wl,-rpath -Wl,/apps/parallel/openmpi/1.10.1/gnu/lib -Wl,--enable-new-dtags -L/apps/parallel/openmpi/1.10.1/gnu/lib -lmpi
```



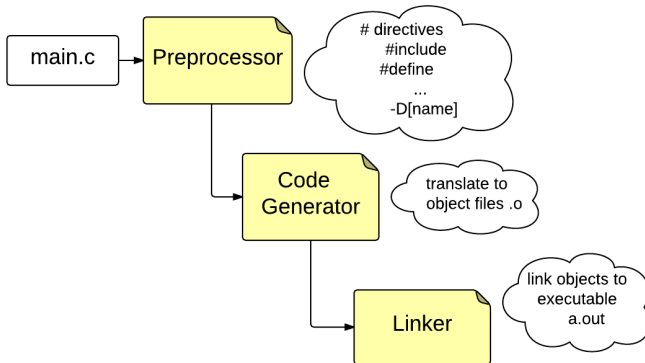
CUDA

module avail cuda

```
----- /apps/modulefiles/compilers -----  
cuda/6.5.14          cuda/8.0.27          cuda/9.1.85  
cuda/7.0.28          cuda/8.0.44  
cuda/7.5.18          cuda/8.0.61(default)
```

- TESLA K40: `nvcc -arch=compute_35 -code=sm_35 ...`
- OpenCL: `-I $CUDAROOT/CL -lOpenCL`

Compile





Compiling a program:

- ➊ **Pre-processing:** #define macros substitution, #include code insertion, remove comments
- ➋ **Compiling:** translate to object files. (machine code collection of symbols referring to variables and functions)
- ➌ **Linking:** build the final executable, link the object files together with any external libraries.

-c	Compile or assemble the source files, but do not link.
-o	Name the outputfile filename.
-g	Produces symbolic debug information.
-D[name]	Predefine [name] as a macro for the preprocessor, with definition 1.
-I[dir]	Specifies an additional directory [dir] to search for include files.
-l[lib]	Search for library when linking.
-static	Force static linkage
-L[dir]	Search for libraries in a specified directory dir.
-fpic	Generate position-independent code.
-version, -v	Show version number.
-help, -h	Show help information, and list flags.
-std=	Conform to a specific language standard



```
main.c:2:17: fatal error: foo.h: No such file or directory
#include "foo.h"
      ^
compilation terminated.
```

Include dir

```
gcc -I/path/to/include main.c
gcc -I$MYPATH main.c
```



Find the paths

```
module show fftw/2.1.5
```

```
-----  
/apps/modulefiles/libraries/fftw/2.1.5:  
  
module-whatis  Enable usage for fftw version 2.1.5  
setenv         FFTWROOT /apps/fftw/2.1.5  
prepend-path   LD_LIBRARY_PATH /apps/fftw/2.1.5/lib  
prepend-path   INFOPATH /apps/fftw/2.1.5/info
```

```
ls $FFTWROOT
```



```
gcc main.c -lfoo
```

```
/usr/bin/ld: cannot find -lfoo  
collect2: error: ld returned 1 exit status
```

```
gcc -L/path/to/lib main.c -lfoo
```

```
./a.out: error while loading shared libraries: libfoo.so: cannot open shared object file: No such file or directory
```

```
echo $LD_LIBRARY_PATH
```

Runtime libraries

```
ldd a.out
```



Optimization Flags

- Without any optimization option, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results
- Turning on optimization flags makes the compiler attempt to improve the performance and/or code size at the expense of compilation time and possibly the ability to debug the program.



Optimization levels common to aris compilers

- O0 No optimization (default), generates unoptimized code but has the fastest compilation time
- O1 Moderate optimization, optimize for size
- O2 Optimize even more, maximize speed
- O3 Full optimization, more aggressive loop and memory-access optimizations
- Ofast Maximizes speed, more aggressive optimizations depending on hardware.

Intel Reference

The O3 (-O3) option is particularly recommended for applications that have loops that do many floating-point calculations or process large data sets. These aggressive optimizations may occasionally slow down other types of applications compared to O2 (-O2).

Leave the compiler to optimize ?

- Example: Dense LU Factorization 1024x1024, gcc

-O0	1.768068 s
-O2	0.569985 s
-O3	0.459217 s
-Ofast	0.517295 s



```
gcc -help=optimizers
icc -help opt
pgcc -help=opt
```

```
The following options control optimizations:
-O<number>      Set optimization level to <number>
-Ofast          Optimize for speed disregarding exact standards
                compliance
-Og             Optimize for debugging experience rather than
                speed or size
-Os            Optimize for space rather than speed
-faggressive-loop-optimizations Aggressively optimize loops using language
                constraints
```



```
gcc -Q -O3 -help=optimizers  
icc -help advanced
```

```
-faggressive-loop-optimizations [enabled]  
-falign-functions [enabled]  
-falign-jumps [enabled]  
-falign-labels [enabled]  
-falign-loops [enabled]  
-fasynchronous-unwind-tables [enabled]  
-fbranch-count-reg [enabled]  
-fbranch-probabilities [disabled]  
-fbranch-target-load-optimize [disabled]  
-fbranch-target-load-optimize2 [disabled]  
-fbtr-bb-exclusive [disabled]
```



GNU Option	Description
-O[0-3]	optimizer level
-Ofast	enables all -O3 optimizations plus -ffast-math, fno-protect-parens and -fstack-arrays
-Os	Optimize for size.
-ffast-math	it can result in incorrect output for programs that depend on an exact implementation of IEEE or ISO rules/specifications for math functions
march=[cputype]	GCC generate code for specific processor: native,ivybridge,core-avx-i,...
mtune=[cputype]	Optimize code for specific processor: native,ivybridge,core-avx-i,... (march=native implies mtune=native)
-Q -march=native – help=target	Show details
-m[target]	Enable use of instructions sets, -mavx, ...
-fomit-frame-pointer	Don't keep the frame pointer in a register for functions that don't need one.
-fp-model[name]	May enhance the consistency of floating point results by restricting certain optimizations.
-fno-alias/-fno-fnalias	Assumes no aliasing(within functions) in the program
-finline-functions	Consider all functions for inlining
-funroll-loops	Unroll loops whose number of iterations can be determined at compile time



Intel Option	Description
-O[0-3]	Optimizer level
-fast	Maximize speed
-Os	Optimize for size
-opt-repot[n]	Generates an optimization report
-x[target]	Generates specialized code for any Intel® processor that supports the instruction set specified by target. AVX,...
-m[target]	Generates specialized code for any Intel processor or compatible, non-Intel processor that supports the instruction set specified by target. AVX,...
-xhost	Generates instruction sets up to the highest that is supported by the compilation host
-parallel	The auto-parallelizer detects simply structured loops that may be safely executed in parallel.
-ip, -ipo	Permits inlining and other interprocedural optimizations
-finline-functions	This option enables function inlining
-unroll, unroll- aggressive	Unroll loops
-[no-]prec-div	Improves [reduces] precision of floating point divides. This may slightly degrade [improve] performance.



PGI Option	Description
-O[0-4]	Optimizer level
-fast	Overall maximize
-Minfo	Display compile time optimization listings.
-Munroll	Uroll loops
-Minline	Inline functions
-Mvect	Vectorization
-Mconcur	Auto-Parallelization
-Mipa=fast,inline	Interprocedural analysis (IPA)



GNU	INTEL	PGI	Description
-O[0-3]	-O[0-3]	-O[0-4]	Optimizer level
-Os	-Os	-	Optimize space
-Ofast	-fast	-fast	Maximizes speed across the entire program.
-mtune,- march=native	-xHost	-	Compiler generates instructions for the highest instruction set available on the host processor. (AVX)
-funroll-loops	-unroll/-unroll- agressive	-Munroll	Unroll loops
-	-opt-streaming- stores	-Mnontemporal	Specifies whether streaming stores are generated
-finline- functions	-ip	-Minline/- Mrecursive	The compiler heuristically decides which functions are worth inlining.
-	-ipo	-Minline Mextract	- Permits inlining and other interprocedural optimizations among multiple source files.



Suggested Flags

GNU

```
gcc -O3 -mavx -march=ivybridge
```

INTEL

```
icc -O3 -xCORE-AVX-I
```

PGI

```
pgcc -O4 -tp=sandybridge
```




Vectorization

- Modern CPUs multiple ops per cycle
- SIMD instructions for operating blocks of data
- SSE: 128 bit (intel core), 4FP single, 2FP double
- AVX: 256 bit (intel Sandy bridge), 8FP single, 4FP double
- AVX2: expansion of the AVX instruction set (intel haswell)
- MIC: 512 bit (intel knights corner), 16FP single, 8FP double



Vectorization Requirements

Requirements

- Vectorisable code,
- Countable loop, constant number of iterations
- only the inner loop is vectorized
- loop must not contain function calls (unless inlined)
- loop must have no dependencies between iterations
- aligned data, unit stride through memory.
- no branches, conditionals

Verify numerical results (FMA instructions)



Vectorization Flags

GNU	INTEL	PGI	Description
-O[2-3], -Ofast	-O[2-3], -fast	-O[2-4], -fast	Enable
-ftree-vectorize	-vec, -simd	-Mvect=simd	Specific enable
-fno-tree-vectorize	-no-vec	-Mnovect	Disable
-march=native	-xHost	-fast	Support AVX
-mavx	-xAVX	-	type of SIMD instruction



Debug I

“Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.”

Programming Problems

- wrong results,
- run-time crashes (segmentation fault)
- pointer errors, array indexes
- memory allocation
- floating point exceptions
- infinite execution



Debug II

- initialization errors
- communication errors , synchronization issues

Debug by hand ?

- insert `printf` in the code
- needs to recompile, hard for complex programs, if parallel?
- debugger allows you to see what is going on *inside* another program while it executes - or what another program was doing at the moment it crashed.



Find the bug

- 1 Understand the whole application
- 2 Identify the test case(s) that cause the fault.
- 3 Run the program until crashes or on specified stop (breakpoint)
- 4 Examine what has happened
- 5 Change things and run again (one bug at a time)



Available Debuggers

Free

- GNU gdb
- module load gdb
- ddd (gui for debuggers)
- module load ddd

Commercial

- INTEL gdb-ia
- PGI pgdbg

Valgrind for memory specific inspection.

<http://doc.aris.grnet.gr/development/debuggers/>



```
gcc [flags] <source file> -o <output file>
```

- `-Wall`, `-Wextra`: Enables warnings for possible errors
- `-Werror`: Converts warnings to errors.
- `-Wfatal-errors`: Breaks compilation from the first warning.

Debug info

```
gcc [flags] -g <source file> -o <output file>
```

- `-g`: default debug information
- `-g``gdb`: produces debugging information specifically intended for `gdb`.



Command "gdb <prog name>" or "gdb"

```
(gdb)
```

```
[nikolouts@login01 ~]$ gdb
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-83.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) █
```

```
(gdb) file <prog name>
```



```
(gdb) help <command>
```

```
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation
```



```
(gdb) help run
```

```
(gdb) help run
Start debugged program.  You may specify arguments to give it.
Args may include "*", or "[...]"; they are expanded using "sh".
Input and output redirection with ">", "<", or ">>" are also allowed.

With no arguments, uses arguments last specified (with "run" or "set args").
To cancel previous arguments and run with no arguments,
use "set args" without arguments.
```



start debugged program

```
(gdb) run
```

```
Program exited normally.  
(gdb) █
```

- Run the code until crash or breakpoint

```
Program received signal SIGSEGV, Segmentation fault.  
_IO_getline_info (fp=0x33b5d8e6c0, buf=0x0, n=1023, delim=10, extract_delim=1,  
of=0x0)  
  at iogetline.c:91  
91          *ptr++ = c;  
(gdb) █
```

- Add breakpoints
- Inspect data



Basic Commands I

```
(gdb) attach
```

attach to a running process outside GDB

```
(gdb) break main.c:6
```

set breakpoint at specified line

```
(gdb) break my_function
```

Set breakpoint at specified function



Basic Commands II

```
(gdb) clear , delete
```

Remove a specified breakpoint

```
(gdb) continue
```

Continue execution after breakpoint

```
(gdb) step
```

Go to the next source line, will step into a function/subroutine



Basic Commands III

```
(gdb) next
```

Go to the next source line, function/subroutine calls are executed without stepping into them

```
(gdb) print <var>
```

Print value of expression (to print array *A@LEN)

```
(gdb) info locals
```

Local variables of current stack frame

Basic Commands IV

```
(gdb) watch <var>
```

Set a watchpoint for an expression. Stops execution whenever the value changes

```
(gdb) list
```

Display (default 10) lines of source surrounding the current lin



Basic Commands V

```
(gdb) backtrace
```

Displays a stack frame for each active subroutine Print backtrace of all stack frames. Use of the 'full' qualifier also prints the values of the local variables.

```
(gdb) core <core>
```

Core files can be examined specifying both an executable program and the core file

snapshot



Basic Commands VI

```
(gdb) generate-core-file
```

Produce a core file from within the GDB session to preserve a snapshot of a program's state

Valgrid: memory leaks

```
(gdb) quit
```

Quit the debugger



```
#include <stdio.h>
#include <stdlib.h>
int myDiv(int, int);
int main(void)
{
  int res, x = 5, y;
  for(y = 1; y < 10; y++){
    res = myDiv(x,y);
    printf("%d,%d,%d\n",x,y,res);
  }
  return 0;
}
int myDiv(int x, int y){
  return 1/( x - y);
}
```

```
PROGRAM main
INTEGER :: myDiv
INTEGER :: res, x = 5, y

DO y = 1, 10
  res = myDiv(x,y)
  WRITE(*,*) x,y,res
END DO
END PROGRAM

FUNCTION myDiv(x,y)
INTEGER, INTENT(IN) :: x, y
myDiv = 1/(x-y)
RETURN
END FUNCTION myDiv
```

```
gcc -g divcrash.c gdb ./a.out
```

```
gfortran -g divcras.f90 gdb ./a.out
```

```
(gdb) run
```

```
Program received signal SIGFPE, Arithmetic exception.  
0x000000000400595 in myDiv ()
```



Intel gdb-ia

- `gdb-ia` Intel Debugger
- debugger, inspect errors: segmentation faults κλπ
- module intel



- ddd Data Display Debugger
- GUI environment
- debug, inspect errors: segmentation faults κλπ
- <http://www.gnu.org/software/ddd/>
- gui vs gdb –tui




ddd <prog name>

```
/* sample.c -- Sample C program to be debugged with DDD */
/* Taken from the DDD Reference documentation */

#include <stdio.h>
#include <stdlib.h>

static void shell_sort(int a[], int size)
{
    int i, j;
    int h = 1;
    do {
        h = h * 3 + 1;
    } while (h <= size);
    do {
        h /= 3;
        for (i = h; i < size; i++)
        {
            int v = a[i];
            for (j = i; j >= h && a[j - h] > v; j -= h)
                a[j] = a[j - h];
            if (i != j)
                a[j] = v;
        }
    }
}
```

Line: "Right Click"

```
I  a = (int *)malloc((argc - 1) * sizeof(int));  
for (i = 0; i < argc - 1; i++)  
    a[i] = atoi(argv[i + 1]);  
  
shell_sort(a, argc);
```




Program -> Run

The screenshot shows the DDD (Data Display Debugger) interface. At the top, there is a menu bar with options: File, Edit, View, Program, Commands, Status, Source, Data, and Help. Below the menu bar is a toolbar with icons for various actions: Lookup, Find, Break, Watch, Print, Display, Plot, Show, Rotate, Set, and Undisp. The main window displays the source code of a C program named 'sample.c'. The code includes `<stdio.h>` and `<stdlib.h>`, and defines a function `shell_sort`. The program is being run with arguments `0 7 5 4 1`. A dialog box titled 'DDD: Run Program (on login02)' is open, showing the arguments `0 7 5 4 1` and a section for 'Run with Arguments' also containing `0 7 5 4 1`. On the right side of the interface, there is a control panel for the debugger, including buttons for 'Run', 'Interrupt', 'Step', 'Stepi', 'Next', 'Nexti', 'Until', 'Finish', 'Cont', 'Kill', 'Up', 'Down', 'Undo', 'Redo', 'Edit', and 'Make'.

```
File Edit View Program Commands Status Source Data Help
(): main
/* sample.c -- Sample C program to be debugged with DDD */
/* Taken from the DDD Reference documentation */

#include <stdio.h>
#include <stdlib.h>

static void shell_sort(int a[], int size)
{
    int i, j;
    int h = 1;
    do {
        h = h * 3 + 1;
    } while (h <= size);
    do {
        h /= 3;
        for (i = h; i < size; i++)
        {
            int v = a[i];
            for (j = i; j >= h && a[j] = a[j - h]; j -= h)
                a[j] = v;
        }
    } while (h != 1);
}

DDD: Run Program (on login02)
Arguments
0 7 5 4 1
Run with Arguments
0 7 5 4 1

DDD (on lo...
Run
Interrupt
Step Stepi
Next Nexti
Until Finish
Cont Kill
Up Down
Undo Redo
Edit Make
```



Debugging OpenMP Applications

- Parallel debug is complex, each process must be debugged simultaneously
- GDB provides basic functionality usually a dedicated debugger
- automatic notification of new threads
- `thread <thread-id>` a command to switch among threads
- `info threads` a command to inquire about existing threads
- thread specific breakpoints



- The GDB thread debugging facility allows you to observe all threads while your program runs—but whenever GDB takes control, one thread in particular is always the focus of debugging. This thread is called the current thread. Debugging commands show program information from the perspective of the current thread.
- When any thread stops (breakpoint) all other threads in the program are also stopped by GDB
- Cannot view values of private and shared variables in parallel regions
- `set scheduler-locking on`: full locking (no thread except the current thread may run). `**else` other threads may execute more than on statement on `gdb step**`



```
gcc -fopenmp -g parsec.c -o parsec
```

```
export OMP_NUM_THREADS=2 gdb parsec (gdb) run
```

```
[Thread debugging using libthread_db enabled]  
Using host libthread_db library "/lib64/libthread_db.so.1"  
".
```

```
[New Thread 0x7ffff7dc8700 (LWP 6544)]
```

```
MASTER: This is thread 0 out of 2
```

```
MASTER: This is thread 1 out of 2
```

```
SECTION B: This is thread 1 out of 2
```

```
SECTION A: This is thread 0 out of 2
```

```
SECTION A: Done
```

```
SECTION B: Done
```

```
[Thread 0x7ffff7dc8700 (LWP 6544) exited]
```

```
[Inferior 1 (process 6540) exited normally]
```



(gdb) break 19

info threads

(gdb) info threads

Id	Target Id	Frame
* 1	Thread 0x7ffff7dc9760	(LWP 24135) "parsec" main. _omp_fn.0 () at parsec.c:19
2	Thread 0x7ffff7dc8700	(LWP 24136) "parsec" main. _omp_fn.0 () at parsec.c:19

```
(gdb) thread apply 1 b 25
```

```
(gdb) thread apply 2 b 35
```

```
(gdb) thread apply all cont
```



Debugging MPI Applications

- invalid arguments, race conditions, deadlocks, ...
- debugging communications "MPIp profiler, Scalasca"
- run with the minimum number of tasks to get the unexpected behavior



Open a debug session for each MPI process

spawn GDB

```
mpirun -n 2 xterm -e gdb a.out
```

gdb (on login01) x	gdb (on login01) x
<pre>GNU gdb (GDB) 7.12.1 Copyright (C) 2017 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "x86_64-pc-linux-gnu". Type "show configuration" for configuration details. For bug reporting instructions, please see: <http://www.gnu.org/software/gdb/bugs/>. Find the GDB manual and other documentation resources online at: <http://www.gnu.org/software/gdb/documentation/>. For help, type "help". Type "apropos word" to search for commands related to "word"... Reading symbols from mpi_hello_world.intelmpi...(no debugging symbols found)...done. (gdb) run Starting program: /users/staff/nikoloutsas/projects/aris_examples/mpi_hello/mpi_hello_world.intelmpi [Thread debugging using libthread_db enabled] Using host libthread_db library "/lib64/libthread_db.so.1". Hello world from processor login01, rank 1 out of 2 processors [Inferior 1 (process 8869) exited normally] (gdb) []</pre>	<pre>GNU gdb (GDB) 7.12.1 Copyright (C) 2017 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "x86_64-pc-linux-gnu". Type "show configuration" for configuration details. For bug reporting instructions, please see: <http://www.gnu.org/software/gdb/bugs/>. Find the GDB manual and other documentation resources online at: <http://www.gnu.org/software/gdb/documentation/>. For help, type "help". Type "apropos word" to search for commands related to "word"... Reading symbols from mpi_hello_world.intelmpi...(no debugging symbols found)...done. (gdb) run Starting program: /users/staff/nikoloutsas/projects/aris_examples/mpi_hello/mpi_hello_world.intelmpi [Thread debugging using libthread_db enabled] Using host libthread_db library "/lib64/libthread_db.so.1". Hello world from processor login01, rank 0 out of 2 processors [Inferior 1 (process 8873) exited normally] (gdb) []</pre>



Valgrind

- Valgrind is an instrumentation framework for building dynamic analysis tools. There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile your programs in detail. You can also use Valgrind to build new tools.
- The most popular of these tools is called **Memcheck**. It can detect many memory-related errors that are common in C and C++ programs and that can lead to crashes and unpredictable behaviour.
- <http://valgrind.org/>

Prepare your program

```
gcc [flags] -g -O0 <source file> -o <output file>
```



Memory Leak

- Memcheck tool
- `-leak-check` option turns on the detailed memory leak detector.

```
#include <stdlib.h>
```

```
void f(void) {  
    int* x = malloc(10 * sizeof(int));  
    x[10] = 0; // problem 1: heap block overrun  
} // problem 2: memory leak -- x not freed
```

```
int main(void) {  
    f();  
    return 0; }
```



```
valgrind --tool=memcheck --leak-check=yes ./a.out
```

```
==18430== Invalid write of size 4
==18430==    at 0x400546: f (leak_check.c:6)
==18430==    by 0x400556: main (leak_check.c:11)
==18430== Address 0x4c29068 is 0 bytes after a block
==18430==    at 0x4A06A2E: malloc (vg_replace_malloc.c:
==18430==    by 0x400539: f (leak_check.c:5)
==18430==    by 0x400556: main (leak_check.c:11)
```



- **Invalid write:** the program wrote to some memory it should not have due to a heap block overrun.
- This one shows that the written memory is just past the end of a block allocated with `malloc()` on line 5 of `leak_check.c`.
- fix errors in the order they are reported, as later errors can be caused by earlier errors



Memory leak messages

```
==18430== 40 bytes in 1 blocks are definitely lost in  
==18430== at 0x4A06A2E: malloc (vg_replace_malloc.c:  
==18430== by 0x400539: f (leak_check.c:5)  
==18430== by 0x400556: main (leak_check.c:11)
```

- **definitely lost:** your program is leaking memory – fix it!
- **probably lost:** your program is leaking memory, unless you're doing funny things with pointers (such as moving them to point to the middle of a heap block).



Uninitialized Variables

```
int var;  
if(var == 0)  
printf("var = 0");
```

```
==19964== Conditional jump or move depends on uninitia  
==19964==      at 0x400534: main (uninit_var.c:6)
```

- difficult to determine the root cause of these errors. Try using the `--track-origins=yes`

```
==22801== Conditional jump or move depends on uninitia  
==22801==      at 0x400534: main (uninit_var.c:6)  
==22801== Uninitialised value was created by a stack  
--22801--      at 0x400528: main (uninit_var.c:4)\texttt
```



Thank you :)