Programming Environment @ LRZ
Parallel Programming of High Performance Systems | 25.02.2019 | Volker Weinberg
Outline

UNIX Intro
- Documentation
- Shells (bash)

Development Tools
- Editors
- Compilers and Linker
- Debuggers
- Performance Tools
- Make
- Revision Control

Parallel Programming
- MPI
- OpenMP
Documentation: man pages

- `man topic`: show manpage about topic
- `man 3 topic`: show topic in section 3 – noted as topic(3)
- `man -a topic`: search topic in all sections and show one after the other
- search man-pages:
  - `man -k topic`
  - `apropos topic`
  - `whatis topic`
Further UNIX documentation

• Short documentation contained in the tools, often by using options like -h, --help, -?

• Info-files of the FSF (/usr/share/info)
  • read via:
    • info topic
    • emacs / Ctrl-h-i
      • select a node with cursor/enter and use
      • n next
      • p previous
      • u 1 level up

• /usr/share/doc/packages/packagename contains documentation for almost every installed rpm package.

• Documentation for module MOD is contained in $MOD_DOC.
Shells

- Interactive interface to the operating system, command interpreter,
- partly very powerful programming language,
- GUI (graphical user interface) vs. CLI (command line interface),
- CLI very useful to automatize work,
- Examples:
  - `bash` - GNU Bourne-Again Shell
  - `ksh` - Public domain Korn shell
  - `tcsh` - C shell
  - `zsh` - the Z shell
  - . . .
- Login Shell can be changed via [https://idportal.lrz.de](https://idportal.lrz.de)
# Development tools

<table>
<thead>
<tr>
<th>Activity</th>
<th>Tools</th>
<th>Linux versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code development</td>
<td>Editors</td>
<td><em>vi, emacs, etc.</em></td>
</tr>
<tr>
<td>Executable creation</td>
<td>Compilers</td>
<td><em>icc, icpc, ifort, gcc, gfortran, g95, pgif90, pgicc</em></td>
</tr>
<tr>
<td>Linking libraries, object files</td>
<td>Linker</td>
<td><em>ln</em></td>
</tr>
<tr>
<td>Archiving</td>
<td>Library Archiver</td>
<td><em>ar</em></td>
</tr>
<tr>
<td>Object file inspection</td>
<td>Object tools</td>
<td><em>ojdump, nm</em></td>
</tr>
<tr>
<td>Debugging</td>
<td>Debuggers</td>
<td><em>gdb, totalview, ddt</em></td>
</tr>
<tr>
<td>Performance analysis</td>
<td>Profilers</td>
<td><em>VTune, Intel Tracing Tools, Scalasca</em></td>
</tr>
<tr>
<td>Automatisation</td>
<td>Make</td>
<td><em>make</em></td>
</tr>
<tr>
<td>Managing software versions</td>
<td>Revision control</td>
<td><em>SVN, git</em></td>
</tr>
<tr>
<td>Integrated development</td>
<td>IDE</td>
<td><em>eclipse</em></td>
</tr>
<tr>
<td>Parallelisation</td>
<td>MPI &amp; OpenMP</td>
<td><em>mpicc, mpiCC, mpif90, -openmp</em></td>
</tr>
</tbody>
</table>
Editors

- **vi**
  - The original code for vi was written by Bill Joy in 1976, as the visual mode for a line editor called *ex* that Joy had written with Chuck Haley.
  - Installed on (almost) every UNIX system per default.
  - Operates in either *insert mode* (where typed text becomes part of the document) or *normal mode* (where keystrokes are interpreted as commands that control the edit session)
  - several implementations available, e.g. Vi Improved (vim)
  - uses lots of *cryptic shortcuts* like `:wq!`
  - THE editor and IDE for UNIX purists.
Editors

• GNU emacs

• Original EMACS was written in 1976 by Richard Stallman and Guy L. Steele, Jr. as a set of Editor MACroS for the TECO editor.
• inofficially: Escape Meta Alt Control Shift
• Very powerful tool which can be used for many things, also for editing!
• Emacs Lisp provides a deep extension capability allowing users and developers to write new commands using a dialect of the Lisp programming language.
• THE editor and IDE for GNU purists.
Editors

I am the psychotherapist. Please, describe your problems. Each time you are finished talking, type RET twice.

How do I use emacs?
I’ll ask the questions, if you don’t mind!

Is there any documentation?
I could ask the same thing myself.

Is it easy to use emacs?
Perhaps you could try to answer that question yourself.

I hate you!
Why do you hate me?
You are useless.

Are you sure that I am useless?
yes.

I see... Well, do you really think this is so?

---xx xdoctorxx Top L1 (Doctor)---
Compiling and linking

- Steps involved in the creation of an executable (C):
Compiling and linking

• Steps involved in the creation of an executable (Intel compiler)

  • `icc –E –lpath-to-includefiles file.c` (Preprocessor)
  • `icc –S file.c` (Compiler, creates assembly in file.s)
  • `icc –c file.s` (Assembler, creates objectfile file.o)
  • `icc file1.o file2.o –llibrary1 –llibrary2 –Lpath-to-libraries … -o exe` (Linker)

• In simple cases all can be done at once by using: `icc file.c -o exe`
Compilers and parallel programming

- A complete list of all compilers and parallel programming libraries available can be obtained using the following module commands at LRZ:

  ```
  module avail -c compilers
  module avail -c parallel
  ```

  which list all packages installed in LRZ's module classes compilers and parallel.
### Compilers

lu65fok@ivy-login:~> module list
Currently Loaded Modulefiles:

<table>
<thead>
<tr>
<th>1) admin/1.0</th>
<th>3) intel/17.0</th>
<th>5) mpi.intel/2017</th>
<th>7) lrz/default</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) tempdir/1.0</td>
<td>4) mkl/2017</td>
<td>6) spack/release/18.2</td>
<td></td>
</tr>
</tbody>
</table>

lu65fok@ivy-login:~>

lu65fok@ivy-login:~> module av

|--------------------- /lrz/sys/share/modules/files/compilers --------------------- |
| bupc/2.26(default) | gcc/5       | intel/17.0(default) | nag/6.2(default) |
| caf/gfortran      | gcc/6       | intel/18.0          | pgi/16          |
| caf/intel(default) | gcc/7(default) | intel/19.0         | pgi/17(default) |
| gcc/4.9           | gcc/8       | java/1.8(default)  | pgi/18          |

|--------------------- /lrz/sys/share/modules/files/parallel --------------------- |
| gpi2/1.3.0         | mpi.intel/2019_gcc | ofi/1.5(default) |
| mpi.intel/2017(default) | mpi.omp/2.0/intel(default) |
| mpi.intel/2017_gcc | mpi.omp/2.1(default) |
| mpi.intel/2018     | mpi.omp/2.1/gcc    |
| mpi.intel/2018_gcc | mpi.omp/2.1/intel(default) |
| mpi.intel/2019     | mpi.omp/3.0/intel  |
Intel Parallel Studio XE

• Recommended development suite for systems with Intel CPUs, e.g. SuperMUC-NG.

• Includes:
  • Intel C/C++ Compiler \((\text{module load intel}; \text{icc}; \text{icpc})\)
  • Intel Fortran Compiler \((\text{module load intel}; \text{ifort})\)
  • Intel Threading Building Blocks (TBB) \((\text{module load tbb})\)
    C++ template library for task parallelism
  • Intel Integrated Performance Primitives (IPP) \((\text{module load ipp})\)
    Building Blocks for Media and Data Applications
  • Intel Math Kernel Library (MKL) \((\text{module load mkl})\)
    Includes highly vectorised and threaded Linear Algebra (BLAS/LAPACK), Fast Fourier Transforms (FFT), Vector Math and Statistics functions.
  • Intel Cilk Plus (extension to the C and C++ languages to support data and task parallelism with just 3 new keywords)
  • Intel OpenMP (shared memory parallel programming)
Most important compiler flags (Intel compiler)

- **-o file**: name output file
- **-O1**: optimize for maximum speed, but disable some optimizations which increase code size for a small speed benefit
- **-O2**: optimize for maximum speed (DEFAULT)
- **-O3**: optimize for maximum speed and enable more aggressive optimizations that may not improve performance on some programs
- **-fast**: enable -xHOST -O3 -ipo -no-prec-div -static
- **-fno-alias**: assume no aliasing in program
- **-g**: produce symbolic debug information in object file (implies -O0)
- **-c**: compile to object (.o) only, do not link
- **-S**: compile to assembly (.s) only, do not link
- **-xAVX**: Generate AVX instructions for SandyBridge
- **-xcore-avx2**: Generate AVX2 instructions for Broadwell
- **-xmic-avx512**: Generate AVX512 instructions for KNL
- **-xcore-avx512**: Generate AVX512 instructions for Skylake

see also http://www.lrz.de/services/compute/supermuc/tuning/intel_options/
Other compilers

- **GNU compilers**
  - Available for various architectures,
  - used to compile the Linux Kernel (even for Xeon Phi),
  - not recommended for Intel systems like SuperMUC,
  - often installed per default on Linux systems under /usr/bin, might be old version) → better load module
  - `module load gcc`
    - `gcc/g++` - GNU project C and C++ compiler
    - `gfortran` - GNU Fortran compiler
Other compilers

• **PGI compiler**

  • developed by “The Portland Group”, part of NVIDIA since 2013, http://www.pgroup.com/
  
  • C/C++ compiler: *module load pgi; pgcc/pgCC*
  
  • Fortran compiler: *module load fortran/pgi; pgfortran*

• PGI CUDA Fortran provides a Fortran version of CUDA to program GPUs and CUDA-x86 Compilers

• PGI Accelerator™ with OpenACC (to program GPUs)
Debuggers

- First, it is strongly recommended you clean and rebuild your code with debugging symbols. For most compilers (C/C++ and Fortran) this is achieved with the `-g` option.

- **gdb – GNU debugger**
  - `module load gdb; gdb`
  - `info gdb`

- command line interface with powerful commands,
- not suited for debugging large MPI jobs.

- GDB's Mascot - *For a fish, the archer fish is known to shoot down bugs from low hanging plants by spitting water at them.*
Debuggers

• Debuggers with graphical Interface (GUI):
  • **Allinea DDT**: Distributed Debugging Tool: a commercial product by Allinea Software.
    • `module load ddt; ddt`
    • PDF: `$DDT_DOC`
    • preferred debugger at SuperMUC, and the largest number of licences is available.
  • **Totalview**: commercial product by RogueWave Software (formerly Etnus, Inc.)
    • `module load totalview; totalview`
    • PDF: `$TOTALVIEW_DOC/*.pdf`
    • can also be used in CLI mode
Intel performance tools

- Intel trace analyser and collector (ITAC)
  - module load itac
  - mpiicc –trace program.c
  - traceanalyser
- Intel Inspector XE – Memory and thread analyser
  - module load inspector_xe
  - inspxe-gui / inspxe-cl
- Intel Vtune Amplifier XE
  - module load amplifier_xe
  - amplxe-gui / amplxe-cl
- Intel Advisor XE
  - module load advisor_xe
  - advixe-gui / advixe-cl
Further performance tools

- **Scalasca** is a software tool measuring and analyzing the runtime behaviour of parallel programs.
  - http://www.scalasca.org/
  - `module load cube scorep`
  - `module load scalasca`

- **LIKWD**: Likwid stands for *Like I knew what I am doing*. This project contributes easy to use command line tools for Linux to support programmers in developing high performance multi threaded programs. Developed by J. Eitzinger et al., RRZE.
  - https://code.google.com/p/likwid/
  - `module load likwid`
Revision control

• SVN
  • http://subversion.apache.org
  • module load subversion
  • developed as a project of the Apache Software Foundation
  • originally designed to be a better CVS.
  • example: svn co http://svn.svnserver.de/svn/<repository>
    svn add file ; svn commit

• GIT
  • http://git-scm.com/
  • module load git
  • https://www.lrz.de/services/software/programmierung/git/
  • Originally written by Linus Torvalds in order to host the Linux kernel.
  • Meanwhile, git is also used by other large projects like Gnome, KDE, Qt, Android, PostgreSQL, and X.org.
  • Falls in the category of distributed source code management tools.

# ls file*.c
# file.c file-v1.c file-v1-afternewdgemm.c file-v1-afternewdgemm2-25.5.2014-backup.c file-v2-final.c file-v2-reallyfinal.c file-v2-reallyfinal-mkl.c
Make

• An utility that automatically builds executable programs and libraries from source code by reading files called `makefiles` which specify how to derive the target program.
• There are now a number of dependency-tracking build utilities, but Make is one of the most widespread.
• GNU Make is the standard implementation of make for Linux systems; http://www.gnu.org/software/make/
Standards-based parallelism

MPI standard

OpenMP standard