# HPC User Environment

Dirk Schmidl

schmidl@rz.rwth-aachen.de

Center for Computing and Communication
RWTH Aachen University
22.03.2010

# Program development tools on Linux

- IDEs
  - eclipse
  - sunstudio
  - kdevelop
- (highlighting) Text editors
  - kate
  - gedit
  - nedit
  - gvim
  - …
- make
- Debuggers
  - TotalView
  - idb
  - …

# make / gmake

“smart” utility to manage compilation of programs and much more

- automatically detects which parts need to be rebuild
- general rules for compilation of many files
- dependencies between files can be handled

Usage:

make <target> or gmake <target>

looks for default makefiles:

- `GNUmakefile`
- `makefile`
- `Makefile`

# make / gmake

Rules:

```
target ... : prerequisites ...
        < tab > command
        < tab > ...
```

target: output file (or only a name)

prerequisites: input files (e.g. source code files)

command: action to be performed

Example:

```
jacobi.exe: jacobi.c main.c
        < tab > gcc jacobi.c main.c -o jacobi.exe
```

# make / gmake

Setting Variables:

1. environment Variables
2. setting variables in the Makefile

   CC = gcc

   CFLAGS = -m64 -O3

3. as argument to make

   $ make jacobi.exe CC=icc

Using Variables:

<tab> $(CC) $(CFLAGS) helloworld.c -o helloworld.exe

# make / gmake

**Recursive Expanded:**
a = $(b) world!
b = Hello

print:
<tab> echo $(a)

echo Hello world!
Hello world!

**Simply Expanded:**
a := $(b) world!
b := Hello

print:
<tab> echo $(a)

echo world!
world!

**expand:**
SRC = jacobi.c
SRC += main.c

# same as SRC = jacobi.c main.c

**if unset:**
CC ?= gcc

# make / gmake

**implicit Variables:**

| | |
|---|---|
| $@ | file name of target |
| $< | name of the first prerequisite |
| $^ | list of all prerequisites |

Example:

```
CFLAGS = -g $(FLAGS_FAST)
SRC = jacobi.c
SRC += main.c
jacobi.exe: $(SRC)
< tab > $(CC) $(CFLAGS) $^ -lm -o $@
```

# make / gmake

Implicit rules:

```
%.o : %.c
< tab > $(CC) -c $(CFLAGS) $< -o $@
```

Example:

```
CFLAGS = -g $(FLAGS_FAST)
OBJS = jacobi.o main.o

jacobi.exe: $(OBJS)
< tab > $(CC) $(CFLAGS) $^ -lm -o $@

%.o: %.c
< tab > $(CC) -c $(CFLAGS) $< -o $@
```

# make / gmake

Phony Targets: name for an commando without associated file

.PHONY: clean build

build: jacobi.exe

clean:
< tab > rm jacobi.exe
< tab > rm jacobi.o main.o

# make / gmake

Errors stop further execution per default.

Ignore returned errors with ‘-’

clean:

< tab > -rm jacobi.exe

< tab > -rm jacobi.o main.o

Suppress printing the command with ‘@’

print:
<tab> echo Hello World!

$ make print
echo Hello World!
Hello World!

print:
<tab> @echo Hello World!

$ make print
Hello World!

# make / gmake

The first target in the makefile is the default target.

Example:

```
CFLAGS = -g $(FLAGS_FAST)
OBJS = jacobi.o main.o

help:
< tab > @echo Use “make build” to build jacobi.exe.
< tab > @echo Use “make clean” to delete jacobi.exe and all object files.

build: jacobi.exe
jacobi.exe: $(OBJS)
< tab > $(CC) $(CFLAGS) $^ -lm -o $@

%.o: %.c
< tab > $(CC) -c $(CFLAGS) $< -o $@
```

# make / gmake

$ make

Use “make build” to build jacobi.exe.

Use “make clean” to delete jacobi.exe and all object files.

# make / gmake

Include other files:

include <filenames>

e.g.: Used to handle sets of variables specified in other files.
Different files for different compilers or platforms.

# make / gmake

Conditions:

*ifeq (var1,var2)*
  *text-if-true*
else
  *text-if-false*
endif

Example:

```
ifeq ($(CC),gcc)
  OPENMP = -fopenmp
else
  OPENMP = -openmp
endif
```

Use $FLAGS_OPENMP on our cluster, set from the module system.

# make / gmake

Useful command line options:

- make target1 target2 : build target1 and target2
- make -n : prints commands which would be executed, but does not execute them
- make -j <njobs> : start <njobs> builds in parallel if possible
- make -C <dir> : change to <dir> before reading the makefile
- make -f <file> : use <file> as makefile

# HPC Program development tips

What are the goals and characteristics of a particular HPC project?

Typical HPC project features:

- (very) long-life time
- portability and
- maintainability and
- efficiency (on available hardware) needed

# HPC Program development tips

At the very beginning think about:

- Will the code be parallelized and to which extent?
- What are and will be the target platforms?
- Choose well-established parallelization paradigm (MPI/OpenMP).
- Choose adequate and well-established language (C/C++/F).
- Use Tools from the very beginning on

  Debugger/ Correctness Checker/ Performance Tools.

If you need advice: servicedesk@rz.rwth-aachen.de

# Virtual Machine

- Virtual Machine is copied on the Laptops
- provides a cluster like environment

Differences:

- modules not useful installed
- Oracle Compiler is default
- Oracle MPI is default

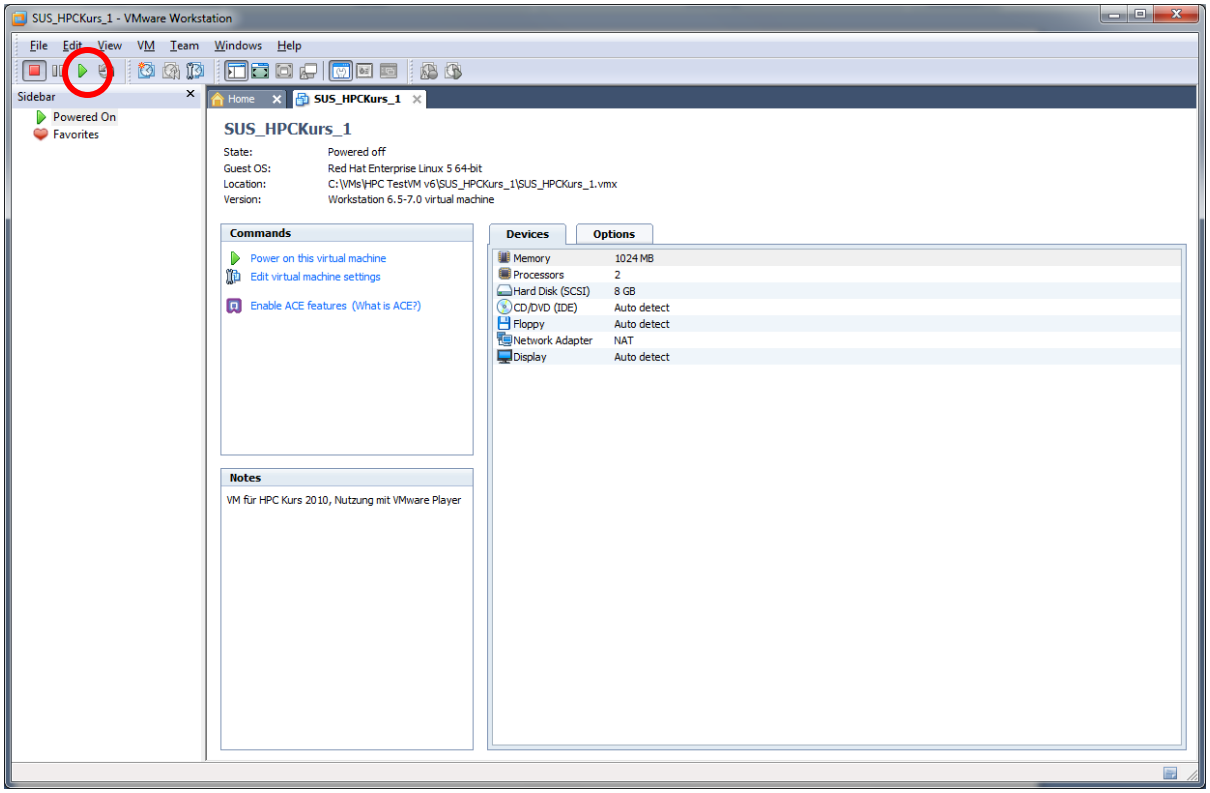
Feature:

- shared folder between VM and Windows

/mnt/hgfs/SharedFolder is C:\SharedFolder

# Virtual Machine

- start VMware Workstation
- click „power on this Virtual Machine“
- user: hpclab
- pw: will be on
  the board



# The end

Thank you for your attention!

Questions?