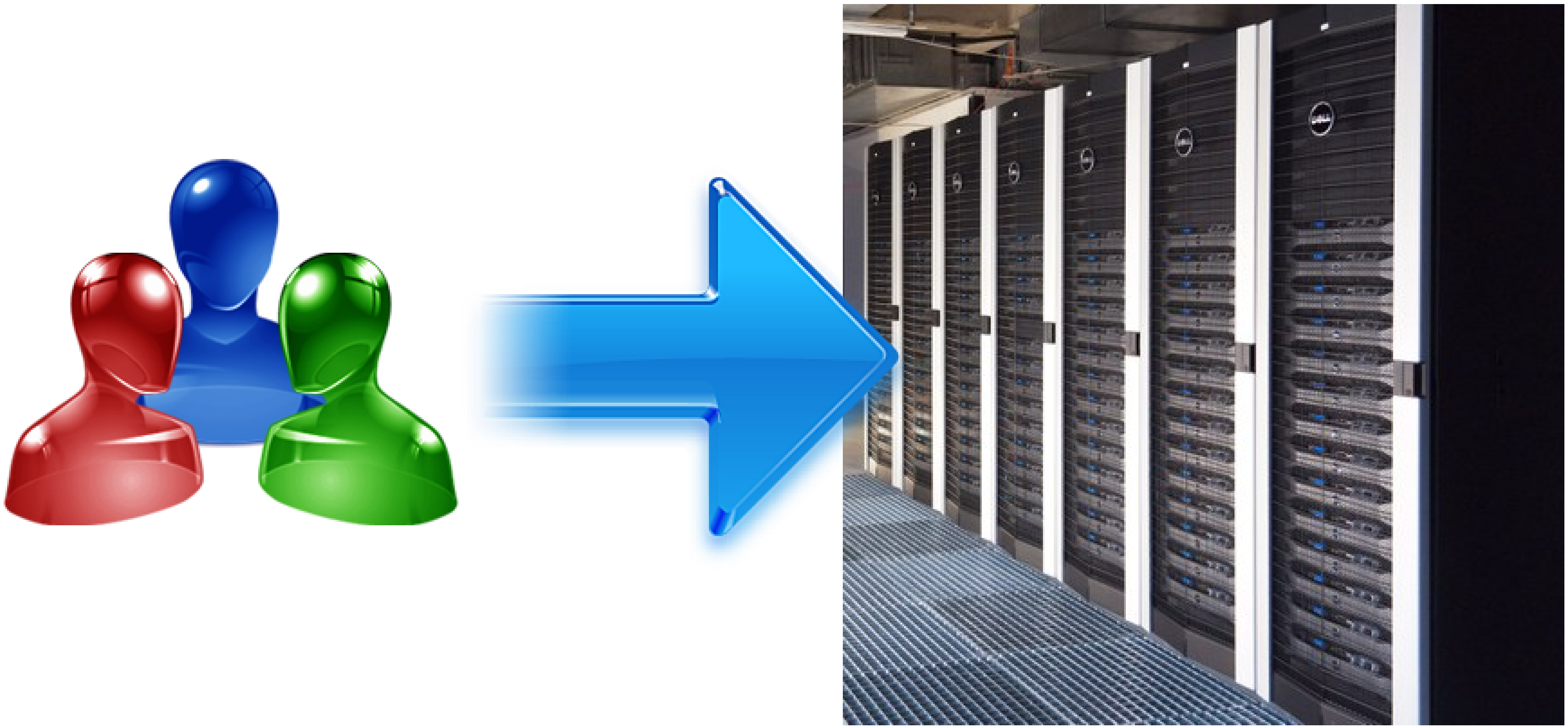
Philipps Universität Marburg

Using the MaRC2 HPC Cluster

HRZ Uni Marburg

Manuel Haim, 06/2013

Using MaRC2???



Using MaRC2 – Overview

- **Get access rights and permissions**
- **Starting a terminal session (Linux, Windows, Mac)**
- **Intro to the BASH Shell (and available directories)**
- **Using the Environment Modules package**
- **Compiling your programs**
- **Submitting jobs to the SUN Grid Engine (SGE)**

Get access rights and permissions

Get access rights and permissions

- **Students / Staff account needed**
- **Ask your workgroup leader if MaRC2 is already being used**

  → he/she must accept the terms of use

  → he/she nominates people who may tell us to grant you access

- **Ask the nominated people at your workgroup to get access**

- **For further questions, terms of use etc., send us an email:**

  → marc@hrz.uni-marburg.de

Starting a terminal session (Linux, Windows, Mac)

Starting a terminal session (Linux, Mac)

- **Start your favourite Terminal application**
- **Connect to MaRC2 via SSH (Secure Shell) by entering:**

```
ssh -p223 username@marc2.hrz.uni-marburg.de
```

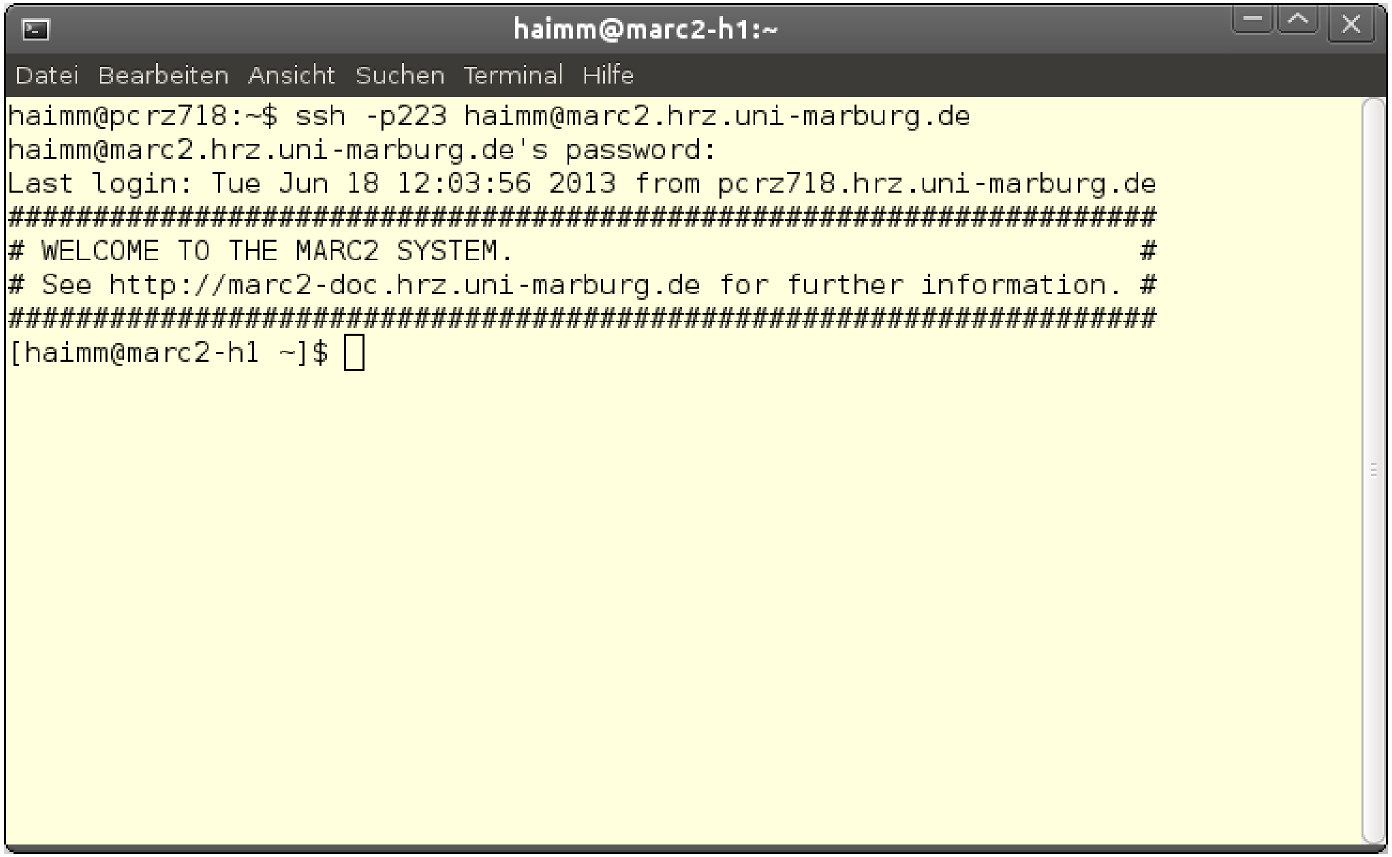
→ replace **`username`** with your Students / Staff account name

→ remember that your account name is case-sensitive!

- **You are then prompted for your account's password:**

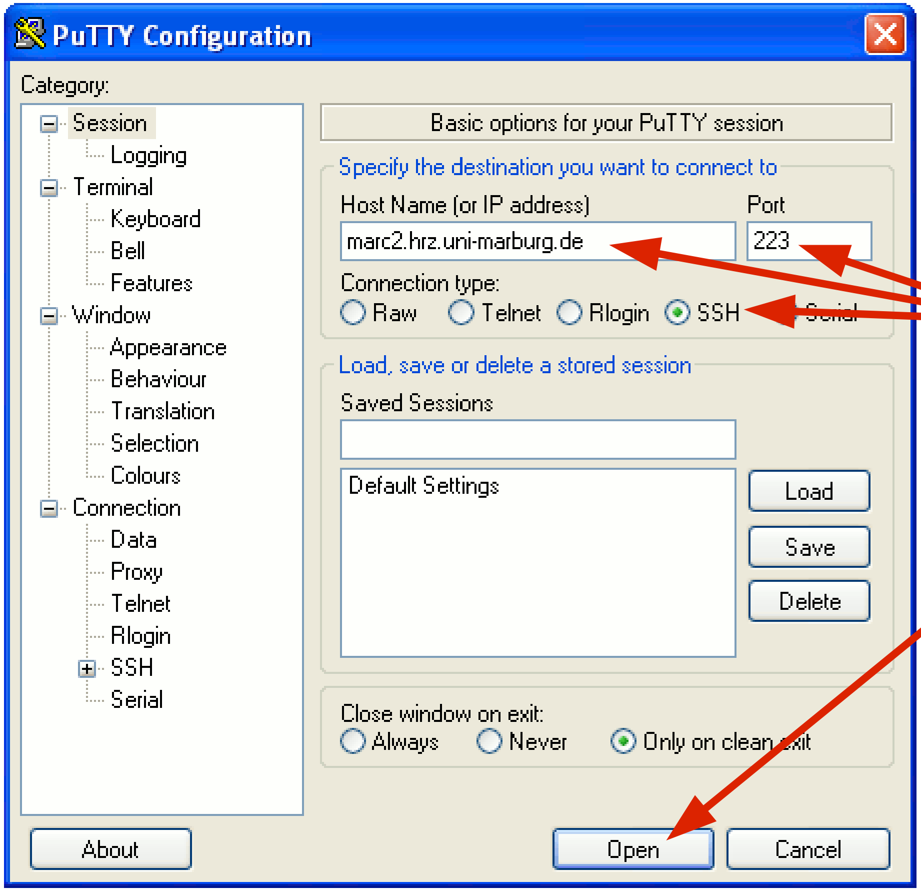
```
username@marc2.hrz.uni-marburg.de's password:
```

Starting a terminal session (Linux) – Example



```
haimm@pcrz718:~$ ssh -p223 haimm@marc2.hrz.uni-marburg.de
haimm@marc2.hrz.uni-marburg.de's password:
Last login: Tue Jun 18 12:03:56 2013 from pcrz718.hrz.uni-marburg.de
####################################################################
# WELCOME TO THE MARC2 SYSTEM.                                     #
# See http://marc2-doc.hrz.uni-marburg.de for further information. #
####################################################################
[haimm@marc2-h1 ~]$ 
```

Starting a terminal session (Windows)



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): marc2.hrz.uni-marburg.de

Port: 223

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

- **Start an SSH client, e.g. PuTTY (www.putty.org)**
- **Select "SSH", then insert host name and port number**
- **Finally, click "open"**
- **A terminal window opens where you will be prompted for username (case-sensitive!) and password**

Some hints

- **To exit the terminal session, just enter `exit` or press Ctrl+D**
- **By default, your login shell is set to `/bin/bash`.**
  You may change this setting globally (and choose another login shell like `csh`, `ksh` or `tcsh`) for your account here:

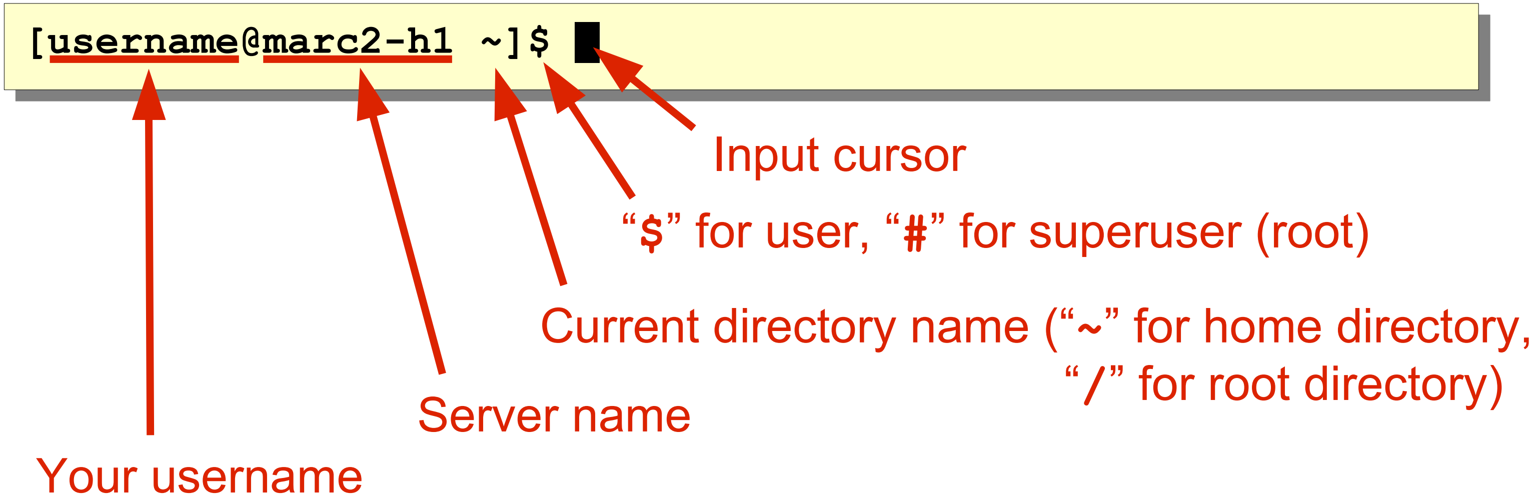
  Staff account:
  https://admin.staff.uni-marburg.de/aendshell.html

  Students account:
  https://admin.students.uni-marburg.de/login_shell.html

Introduction to the BASH Shell (and available directories)

Introduction to the BASH Shell (Bourne-again shell)

- **Right after login, the BASH Shell is started by default and shows a command prompt like this:**



```
[username@marc2-h1 ~]$ █
```

Input cursor

“**$**” for user, “**#**” for superuser (root)

Current directory name (“**~**” for home directory,
“**/**” for root directory)

Server name

Your username

- **To exit, simply enter the command `exit` or press Ctrl+D**

First steps with BASH-builtin commands

- `pwd` – print current (working) directory:

```
[username@marc2-h1 ~]$ pwd
/home/username
```

- `cd` – change current directory:
  `cd ..` – go to parent directory
  `cd <dirname>` – go to subdirectory

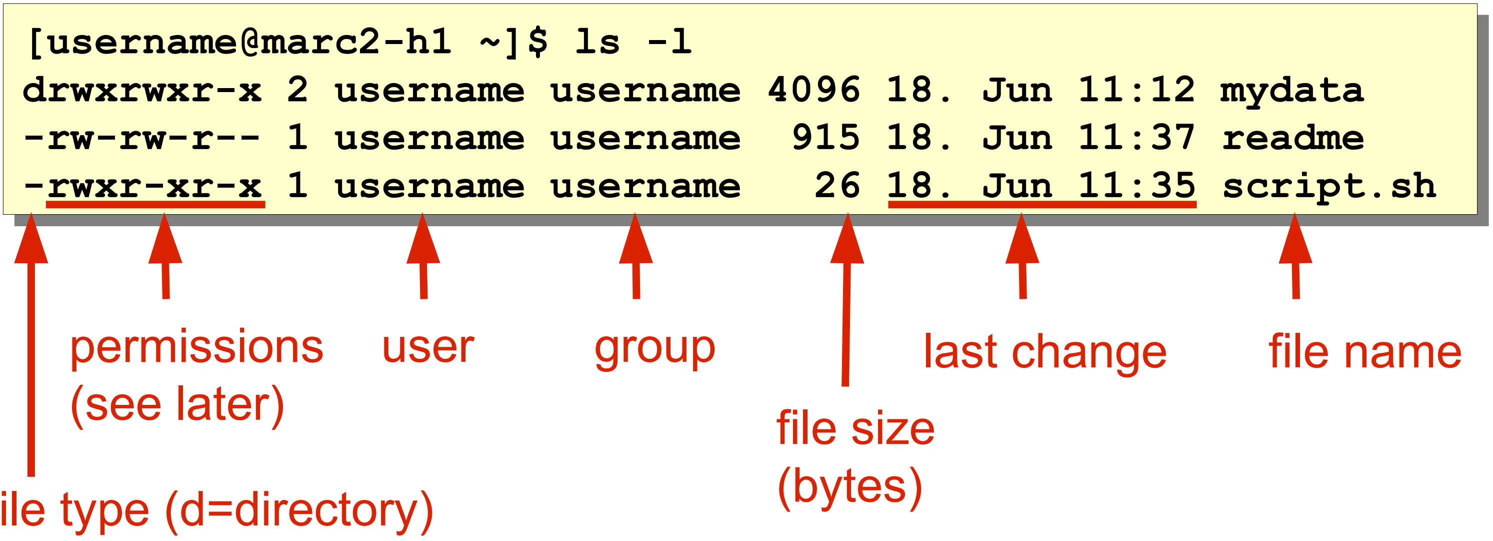
```
[username@marc2-h1 ~]$ cd ..
[username@marc2-h1 home]$ pwd
/home
[username@marc2-h1 home]$ cd username
[username@marc2-h1 ~]$ pwd
/home/username
```

Additional commands

- `ls` – list directory contents:

```
[username@marc2-h1 ~]$ ls
mydata   readme   script.sh
```

`ls -l` – list contents in long format



```
[username@marc2-h1 ~]$ ls -l
drwxrwxr-x 2 username username 4096 18. Jun 11:12 mydata
-rw-rw-r-- 1 username username  915 18. Jun 11:37 readme
-rwxr-xr-x 1 username username   26 18. Jun 11:35 script.sh
```

permissions
(see later)

user

group

file size
(bytes)

last change

file name

file type (d=directory)

Additional commands (2)

- **`ls -a` – show all files (even hidden ones):**

```
[username@marc2-h1 ~]$ ls -a
.    .bash_history  .bash_profile  mydata  script.sh
..   .bash_logout   .bashrc        readme
```

(files starting with “.” are usually not shown)

Additional commands (3)

- `mkdir <dirname>` – make subdirectory:

```
[username@marc2-h1 ~]$ mkdir temp
[username@marc2-h1 ~]$ cd temp
[username@marc2-h1 temp]$ pwd
/home/username/temp
```

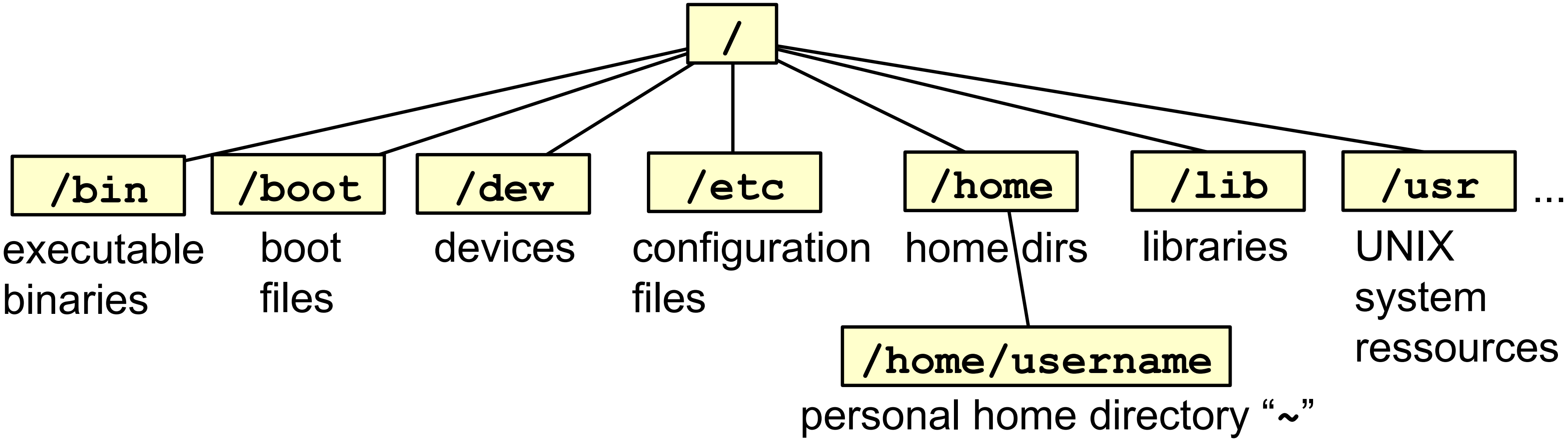
- `rmdir <dirname>` – remove directory (must be empty):

```
[username@marc2-h1 temp]$ cd ..
[username@marc2-h1 ~]$ rmdir temp
```

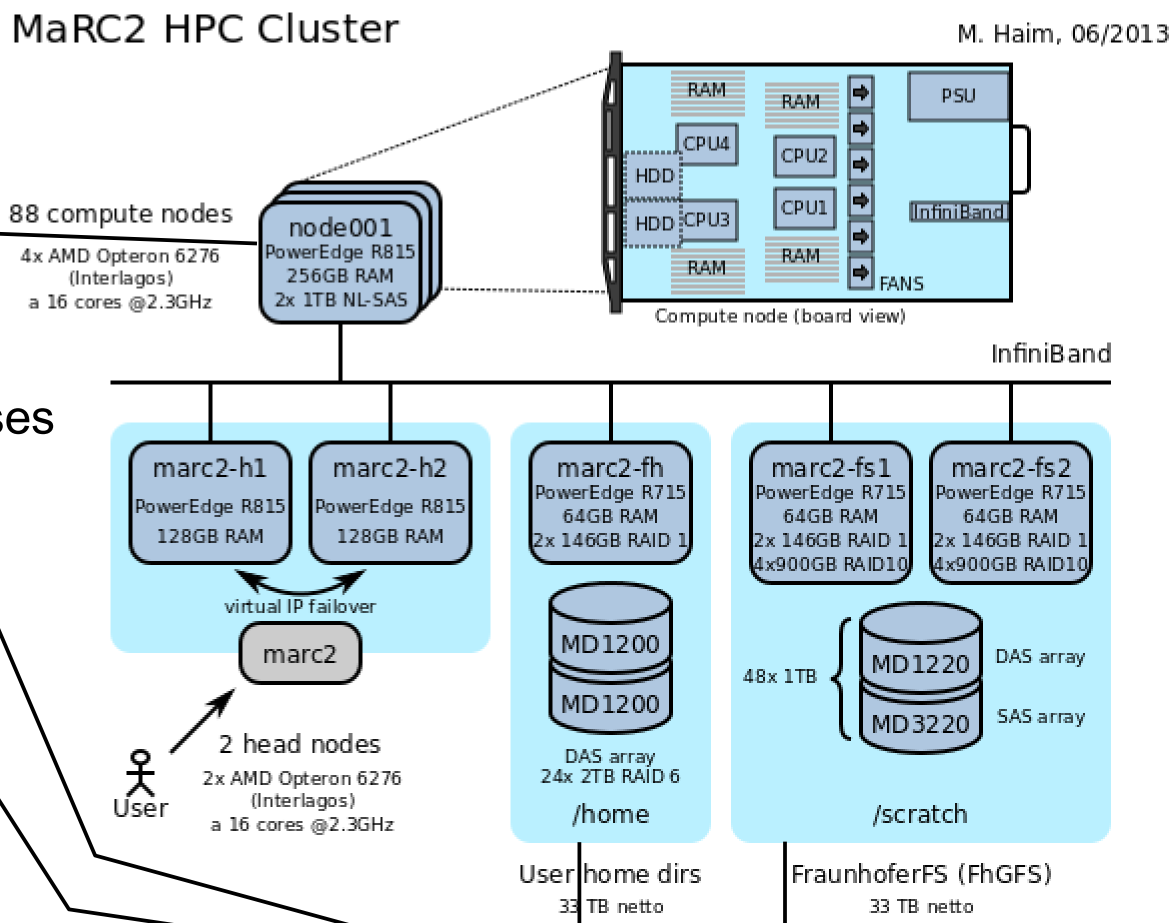
- `rm <filename>` – remove (delete) file
  `rm -R <dirname>` – remove (delete) directory recursively!
  `Rm -Rf <dirname>` – just as above, but “force”, don't ask ;-)

Available directories on a typical Linux system

- **The Unix/Linux filesystem is organized in a tree-like structure**
- **The top-most directory "/" is called the root directory**
- **All filesystem entries (including mounted filesystems) are branches of this root.**



Some special directories on MaRC2



/local

local storage (1.8TB)
on node001 – node088,
available to local processes
(not shared via network)

```
mkdir /local/username
```

/home/username

personal home directory "~"

/home/groupname

shared group directory

/home/software

shared software

/home

shared home dirs (33TB),
optimized for integrity

/scratch

shared directory (33TB),
optimized for speed

```
mkdir /scratch/username
```

Getting help

- `man <command>` – **Show man(ual) page for command:**

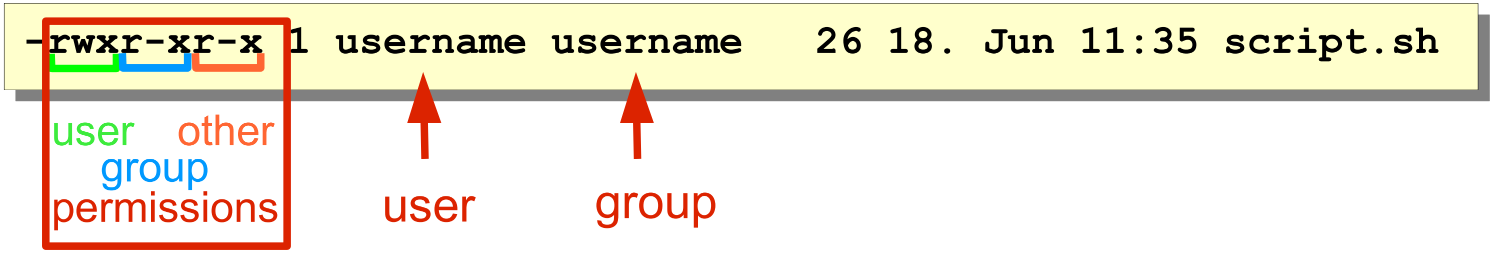
```
[username@marc2-h1 ~]$ man ls
```

→ Press space for next page, b for previous page
→ Arrow keys and PgUp/PgDn will also work nowadays
→ Press q to quit

- `<command> --help` **or** `<command> -h` – **Show help**
  (not supported by all commands)

Change file owner and permissions

- **This is what `ls -l` shows:**



```
-rwxr-xr-x 1 username username   26 18. Jun 11:35 script.sh
```

r = read
w = write
x = execute

- **`chmod <permissions> <file>` – Change permissions**
  example: **`chmod u=rwx,g=rx,o=rx script.sh`**

- **`chown <user>:<group> <file>` – Change owner and group**

Some tips and tricks

- **Select text to copy, then paste with middlemouse button**
  (PuTTY: Use right mouse button)
- **Tab auto-completes your input**
- **Shift+PgUp/PgDn scrolls the screen buffer up and down**
  (maximum scrollback buffer size can be set in your terminal options)
- **ArrowUp and ArrowDown allows you to navigate through your former input**
- **Ctrl-R allows you to reverse-i-search in your former input**
- **Edit your .bashrc and .bash_profile for personal settings**

Using the Environment Modules package

Using the Environment Modules package – why?

- **Each piece of software typically needs a set of environment variables to be set, in order to run properly**
  → e.g. the $PATH variable which points to the bin directories
- **On a HPC cluster like MaRC2, there may be many different toolsets and toolset versions with the same purpose and/or the same command names → Conflict!**
- **With the Environment Modules package, you can dynamically modify your environment by loading so-called “modulefiles”**
  → Use our predefined modulefiles, no need to create your own :-)

Using the Environment Modules package – how?

- **`module avail` – show available modules**

```
[haimm@marc2-h1 ~]$ module avail

------------------------ /usr/share/Modules/modulefiles ------------------------
dot         module-cvs  module-info modules     null        use.own

--------------------------- /usr/share/ModulesLocal ----------------------------
acml/gfortran-5.1.0(default)           openmpi/pgi/1.4.3-qlc
acml/gfortran-5.2.0                    openmpi-1.6.3/gcc/1.6.3
acml/ifort-5.1.0 (AMD Core Math Library) openmpi-1.6.3/icc/1.6.3
acml/pgi-5.1.0                         openmpi-1.6.3/pgi/1.6.3
gcc/4.4.6                              parastation/mpi2-gcc-5.0.27-1(default)
gcc/4.6.2(default)                     parastation/mpi2-gcc-mt-5.0.27-1
gcc/4.7.2                              parastation/mpi2-intel-5.0.27-1
intel/intelPSXE2011SP1-32              parastation/mpi2-intel-mt-5.0.27-1
intel/intelPSXE2011SP1-64              parastation/mpi2-pgi-5.0.27-1
intel/intelPSXE2013SP1-64              parastation/mpi2-pgi-mt-5.0.27-1
openmpi/gcc/1.4.3-qlc                  pgi/12.2(default)
openmpi/intel/1.4.3-qlc                pgi/13.2 (Portland Group Inc. Compiler)
```

- **`module load <modulefile>` – load module**

  → You do not need to specify the full path in order to load the default, e.g.:

```
[haimm@marc2-h1 ~]$ module load gcc
```

Using the Environment Modules package – how (2)

- **`module list` – show loaded modules**

```
[haimm@marc2-h1 ~]$ module list
Currently Loaded Modulefiles:
  1) acml/gfortran-5.1.0             3) parastation/mpi2-gcc-5.0.27-1
  2) gcc/4.6.2
```

- **`module unload <modulefile>` – unload module**
  → Again, you do not need to specify the full path:

```
[haimm@marc2-h1 ~]$ module unload gcc
```

- **More information: `man module`**

Compiling your programs

Compiling your programs – Choose a compiler

- **Several compilers available:**
  → **`gcc`** – GNU C Compiler
  → **`gfortran`** – GNU Fortran Compiler
  → **`icc`** – Intel C Compiler
  → **`ifort`** – Intel Fortran Compiler
  → **`pgcc`** – Portland Group Inc. (PGI) C Compiler
  → **`pgfortran`** – Portland Group Inc. (PGI) Fortran Compiler

- **MPI Compilers (OpenMPI or Parastation MPI, GNU/Intel/PGI):**
  → **`mpicc`** – MPI C Compiler
  → **`mpicxx`** – MPI C++ Compiler
  → **`mpif77`** – MPI Fortran77 Compiler

A simple C program

- **Create a file `hello-world.c` with the following content:**

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

int main() {
   printf("Hello World!\n");
   exit(0);
}
```

- **Compile using `gcc`:** (remember to have the module loaded first!)

```
[haimm@marc2-h1 ~]$ gcc hello-world.c -o hello-world
```

- **Then test locally:** (don't do this with huge complex programs!)

```
[haimm@marc2-h1 ~]$ ./hello-world
Hello World!
```

We could also use a `Makefile`

- **Create a file `Makefile` with the following content:**

You must insert a tab here!

```
PROG=hello-world
SRC=hello-world.c
CC=gcc

all: clean build

clean:
	rm -f $(PROG)

build:
	$(CC) $(SRC) -o $(PROG)

run: all
	./$(PROG)
```

- **Now just enter `make` or `make run` to compile/run your code**

Submitting jobs to the SUN Grid Engine (SGE)

SUN Grid Engine???

- **The SUN Grid Engine (SGE) implements a queueing system**
- **Users can submit jobs to the SGE queues and define boundary conditions (like runtime, memory etc.)**
- **The SGE permanently reviews the available resources, evaluates pending jobs and executes them on the compute nodes**
- **The SGE is open source software with a free-to-use license**
  → Its successor, the Oracle Grid Engine, is only available commercially

How to submit a new job to the SGE

- **First, write a start script for your software:** (e.g. `hello-world.sh`)

```
#!/bin/bash

#$ -S /bin/bash
#$ -e ./stderr-file
#$ -o ./stdout-file
#$ -l h_rt=10

echo "hello-world running on host $(hostname)"

./hello-world

exit 0
```

- **Then submit your start script:** (`-cwd`: execute from current working dir)

```
$ qsub -cwd hello-world.sh
Your job 144406 ("hello-world.sh") has been submitted
```

Get information on your submitted jobs

- `qstat` – **List all your jobs**

```
$ qstat
job-ID  prior   name       user     state submit/start at     queue     slots ja-task-ID
---------------------------------------------------------------------------------------
 144407 0.00000 start.sh   haimm    qw    06/19/2013 12:43:26                 1
```

SGE may take a few seconds
to give your job a non-zero priority

q=queued, w=waiting,
E=Error, r=running

1 CPU slot (core)

- `qstat -u "*"` – **List all jobs from all users**

- `qstat -j <job-id>` – **Get information on a single job**

```
$ qstat -j 144406
==============================================================
job_number:                 144406
exec_file:                  job_scripts/144406
submission_time:            Wed Jun 19 12:40:43 2013
owner:                      haimm
uid:                        30376
...
```

Get information on your submitted jobs (2)

- **`qalter -w v <job-id>` – Get scheduler information for job** (may be helpful if your job won't run)

```
$ qalter -w v 144406

Job 144406 queue instance "parallel_long@node028.marc2" dropped because it is
temporarily not available

Job 144406 queue instance "parallel_long@node032.marc2" dropped because it is
temporarily not available

Job 144406 queue instance "parallel_long@node043.marc2" dropped because it is
temporarily not available

...
```

- **`qdel <job-id>` – Delete (terminate) a submitted job**

Hint for Java users (memory boundaries)

- **Java usually takes more memory than you think it takes**
- **SGE's `h_vmem` value must be larger than Java's `-Xmx` value, you may test it with a simple “Hello world” program like this:**

```
#!/bin/bash

#$ -S /bin/bash
#$ -e ./stderr-file
#$ -o ./stdout-file
#$ -l h_vmem=9G, h_rt=60

java -Xms5120m -Xmx5120m -cp ./Hello.jar Hello

exit 0
```

Additional resources

- MaRC2 wiki:
  http://marc2-doc.hrz.uni-marburg.de
  (see Users Guide for information on the SGE, MPI and a usage tutorial)
- MaRC2 info page (German):
  http://www.uni-marburg.de/hrz/infrastruktur/zserv/cluster
- Further questions?
  → Just ask :-)
  → or send us an email: marc@hrz.uni-marburg.de

References:
Several icons within this presentation are taken from “Crystal Project” (GNU LGPL).