LSF (Platform Load Sharing Facility) Basics

By Sumit Ghosh and Kumar Aakash

Graduate Assistants in Research Computing

Pursuing MS in Computer Science at Northeastern University
The `bqueues` command displays information about queues. Here is a sample output:

```
[ghosh.s@discovery2 ~]$ bqueues

<table>
<thead>
<tr>
<th>QUEUE_NAME</th>
<th>PRIOR</th>
<th>STATUS</th>
<th>MAX</th>
<th>JL/U</th>
<th>JL/P</th>
<th>JL/H</th>
<th>NJOBS</th>
<th>PEND</th>
<th>RUN</th>
<th>SUSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel-ib</td>
<td>40</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12096</td>
<td>11264</td>
<td>832</td>
<td>0</td>
</tr>
<tr>
<td>ser-par-10g</td>
<td>40</td>
<td>Open:Active</td>
<td>-</td>
<td>512</td>
<td>-</td>
<td>-</td>
<td>840</td>
<td>40</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>interactive-10g</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hadoop-10g</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>interactive-ib</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ser-par-10g-2</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>580</td>
<td>-</td>
<td>-</td>
<td>844</td>
<td>0</td>
<td>844</td>
<td>0</td>
</tr>
<tr>
<td>ser-par-10g-3</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>880</td>
<td>-</td>
<td>-</td>
<td>1090</td>
<td>0</td>
<td>1090</td>
<td>0</td>
</tr>
<tr>
<td>ondrechen-10g</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>largemem-10g</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ht-10g</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>par-gpu</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>768</td>
<td>-</td>
<td>-</td>
<td>5984</td>
<td>5056</td>
<td>928</td>
<td>0</td>
</tr>
<tr>
<td>redwood</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8980</td>
<td>8340</td>
<td>640</td>
<td>0</td>
</tr>
<tr>
<td>maloney</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>suh</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>plizer</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fu</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>krioukov</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>lhtct3alv</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hanchen</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>west</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3860</td>
<td>3660</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>westlargetmem</td>
<td>30</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>22</td>
<td>0</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>
```
bqueues -l
Displays queue information in a long multiline format.

The queue

The hosts for this queue
bhosts
Displays hosts and their static and dynamic resources

```
[ghosh.s@discovery2 ~]$ bhosts

<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>J/L/U</th>
<th>MAX</th>
<th>NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-0-000</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-001</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-002</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-003</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-004</td>
<td>ok</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>compute-0-005</td>
<td>ok</td>
<td>-</td>
<td>32</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-006</td>
<td>ok</td>
<td>-</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-007</td>
<td>ok</td>
<td>-</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-008</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-009</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-010</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-011</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-012</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-013</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-014</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-015</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-016</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-017</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-018</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-019</td>
<td>ok</td>
<td>-</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-020</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>compute-0-021</td>
<td>closed</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
**bhosts** `<host_group>`

Displays hosts and their static and dynamic resources for the group
bsub
Submits a job to LSF

• 2 types of jobs – batch and interactive

• Batch:
  • Create a submit script containing information about your job and commands to the LSF
  • Submit the script to LSF using: `bsub < submit_script`

• Interactive:
  • Request LSF for allocating a node to run interactive jobs with
    `bsub –Is –XF –n <total_no_of_cores> -q <queue_name>
    -R span[ptile=<no_of_cores_in_a_node>] /bin/bash`
  • Start your interactive job in the allocated node (not on the login node)
**bsub**

For interactive jobs

- `bsub -Is -XF -n <total_no_of_cores> -q <queue_name> -R span[ptile=<no_of_cores_in_a_node>] /bin/bash`

**Where:**

- `-Is` : Interactive terminal/shell
- `-XF` : X11 forwarding
- `-n` : to specify the total number of compute cores needed
- `-q` : to specify the queue-name from where to get the nodes
- `-R` : Resource request specifying no. of cores in a compute node
- `/bin/bash` : the shell to use
bsub
For interactive jobs

```
bsub -Is bqueues -l ser-par:10g-2
```

```
[ghosh.sgdiscovery2 @j5 bhosts nodes10g2]
HOST_NAME  STATUS     3J/U     MAX NJOBS  RUN SSUSP USUSP RSV
compute-0-066  closed    40  40    40      0      0      0      0
compute-0-067  closed    40  40    40      0      0      0      0
compute-0-068  ok        40  0      0      0      0      0      0
compute-0-069  closed    40  40    40      0      0      0      0
compute-0-070  closed    40  40    40      0      0      0      0
compute-0-071  closed    40  40    40      0      0      0      0
compute-0-072  ok        40  5      5      0      0      0      0
compute-0-073  closed    40  40    40      0      0      0      0
compute-0-074  closed    40  40    40      0      0      0      0
compute-0-075  closed    40  40    40      0      0      0      0
compute-0-076  closed    40  40    40      0      0      0      0
compute-0-077  closed    40  40    40      0      0      0      0
compute-0-078  closed    40  40    40      0      0      0      0
compute-0-079  closed    40  40    40      0      0      0      0
compute-0-080  closed    40  40    40      0      0      0      0
compute-0-081  closed    40  40    40      0      0      0      0
compute-0-082  closed    40  40    40      0      0      0      0
compute-0-083  closed    40  40    40      0      0      0      0
compute-0-084  closed    40  40    40      0      0      0      0
compute-0-085  closed    40  40    40      0      0      0      0
compute-0-086  closed    40  40    40      0      0      0      0
compute-0-087  closed    40  40    40      0      0      0      0
compute-0-088  closed    40  40    40      0      0      0      0
compute-0-089  closed    40  40    40      0      0      0      0
compute-0-090  ok        40  0      0      0      0      0      0
compute-0-091  closed    40  40    40      0      0      0      0
compute-0-092  closed    40  40    40      0      0      0      0
compute-0-093  closed    40  40    40      0      0      0      0
compute-0-094  ok        40  0      0      0      0      0      0
compute-0-095  closed    40  40    40      0      0      0      0
```

The resource allocated to the job is `compute-0-072`.


**bjobs**
Displays and filters information about LSF jobs

**bjobs** –w

- Wide format. Displays job information without truncating fields.
**bjobs**
Displays and filters information about LSF jobs

**bjobs –u all**

- Only displays jobs that have been submitted by the specified users or user groups.
- The keyword all specifies all users.
**lsload** `<host_name>`
Displays load information for hosts

```
[ghosh.s@discovery2 ~]$ lsload compute-0-001
HOST_NAME    status  r15s  r1m  r15m  ut   pg  ls  it   tmp    swp   mem
compute-0-001  ok   16.2  16.0  16.0  50%  0.0  0  30448  776G  3.9G  121.5G
[ghosh.s@discovery2 ~]$  
```
Sample Submit Script for Batch jobs

nroy@discovery4 mpi4py-test]$ cat bsub_mpi4py.bash
#!/bin/sh
#BSUB -J JOB.1
#BSUB -o output_file
#BSUB -e error_file
#BSUB -n 8
#BSUB -q ht-10g
#BSUB -cwd /home/nroy/mpi4py-test
####### THIS IS A TEMPLATE FILE FOR TCP ENABLED MPI RUNS ON THE DISCOVERY CLUSTER
########
#### #BSUB -n has a value equal to the given value for the -np option ####
# prefix for next run is entered below
# file staging code is entered below

####### Enter your working directory below - this is the string returned from issuing the command  ####
"pwd"
# IF you stage your files this is your run directory in the high speed scratch space mounted
# across all compute nodes
work=/home/nroy/mpi4py-test

############### DO NOT EDIT ANYTHING BELOW THIS LINE###############
cd $work
tempfile1=hostlistrun
tempfile2=hostlist-tcp
echo $LSB_MCPU_HOSTS > $tempfile1
declare -a hosts
read -a hosts < ${tempfile1}
for ((i=0; i<$#hosts[@]; i+= 2)) ;
do
HOST=${hosts[$i]}
CORE=${hosts[(($i+1))]};
echo $HOST:$CORE >> $tempfile2
done
############### DO NOT EDIT ANYTHING ABOVE THIS LINE###############

tmpfile1=hostlistrun
tmpfile2=hostlist-tcp
rm $work/$tempfile1
rm $work/$tempfile2

########## Change only the -np option giving the number of MPI processes and the executable to use
########## DO NOT CHANGE ANYTHING ELSE BELOW FOR mpirun OPTIONS
########## MAKE SURE THAT THE "#BSUB -n" is equal to the "-np" number below. IN this example it is 8.

mpirun -np 8 -prot -TCP -lsf helloworld.py
#/home/nroy/mpi4py-test/helloworld.py
# any clean up tasks and file migration code is entered below

[nroy@discovery4 mpi4py-test]$
Consider scenarios...

• Need to submit 1000 jobs...
  • Make 1000 submit scripts
  • Make different locations for input and output... so on...

• Need to ensure consecutive execution rather than parallel execution of jobs...
  • Need to submit jobs 1 at a time... what if a large number of jobs are there??
We can AUTOMATE .... !!!
Consider scenarios...

• Need to submit 1000 jobs...
  • Make 1000 submit scripts
  • Make different locations for input and output... so on...
  • Script will do it 1000 times...

• Need to ensure consecutive execution rather than parallel execution of jobs...
  • Need to submit jobs 1 at a time... what if a large number of jobs are there??
  • Script will provide dependency information...
Automating Job Submissions

• Technologies used: bash, perl or python scripting languages.

• Things to take care of before automating the batch job submissions:
  
  • How much data does your job process in 24 hours?
  
  • How much output does your job generate in 24 hours?
  
  • How many different jobs should you break your complete process in to complete quickly?
  
  • Are the jobs to be run in serial (i.e. the next job depends on the previous one)
  
  • Are the jobs to be run in parallel (i.e. the job does not depend on the previous one)
The submit script

```
#!/bin/sh
#BSUB -J JOB.1
#BSUB -D output_file
#BSUB -e error_file
#BSUB -n 8
#BSUB -q ht-10g
#BSUB -w /home/nroy/mpi4py-test

######## THIS IS A TEMPLATE FILE FOR TCP ENABLED MPI RUNS ON THE DISCOVERY CLUSTER ########
### #BSUB -n has a value equal to the given value for the -np option ###
# prefix for next run is entered below
# file staging code is entered below
#### Enter your working directory below
work=/home/nroy/mpi4py-test

# any cleanup tasks and file migration code is entered below
# any clean up tasks and file migration code is entered below

cd $work

tempfile1=`hostlistrun`
tempfile2=`hostlist-tcp`

for ((i=0; i<${#hosts[@]}; i += 2)) ;
do
    HOST=${hosts[$i]}
    CORE=${hosts[(($i+1))]}
    echo $HOST:$CORE >> $tempfile2
done

mpirun -np 8 -prot TCP -lsf helloworld.py
```

- **Used to submit a single job**
- **Options to set various parameters while job submission**
  - `-J`: Specifies the job name
  - `-D`: Specifies a file where the output from the job shall be stored
  - `-e`: Specifies a file where the errors from the job shall be stored
  - `-n`: Specifies the number of cores that the jobs needs to use
  - `-q`: Specifies the queue to which the job needs to be submitted
  - `-cwdb`: Specifies the current working directory
  - `-work`: Specifies where the input and output files need to be generated
  - `-w`: Specifies to wait for another job with job name

- **mpirun command**: Starts jobs on several devices.
  - `-np`: Specifies the number of processes to run
  - `-hostfile`: Specifies the specific hosts on which to run processes
  - `-prot`: Protocol to use for the communication
  - `-lsf`: Use LSF for running processes on nodes
submit.py

Start:

1. Read the number of jobs to divide the process in.

2. Create a different folder and a different submit script for each job.

3. Adds “#BSUB –w” option in each of the submit script to make a job wait for previous one to complete.

4. Submits all the jobs created in this manner.

End.
How does the automation work?

- LSF generates the input and output files in the working directory mentioned in the submit script.
- LSF recognizes each job using the job name specified in the submit script.
- Automation takes advantage of these facts to create multiple jobs for a given process.
- The automation may use a template submit script to build the other scripts from (the example script does)
- The –w option in the submit script allows for a job to be started based on the exit status of another job.
- Other options like –E and –K can be used to provide a conditional dependency on another job.
- In the end creating this automation is rather simple.
What we came up with? We can do better ...

```python
import os
dir_prefix = "job_"
number_of_jobs = 5

def write_new_submit_script(job_number):
    ""
    Read the template.bash file line by line
    and create new submit scripts.
    ""
    job_name = dir_prefix + str(job_number)
template = open("template.bash").readlines()

    new_submit_script = open(job_name + "" + job_name + ".bash", "w");

    for line in template:
        newline = line

        if "#BSUB -J" in line:
            newline = "#BSUB -J " + job_name + "\n"

        if "#BSUB -cwd" in line:
            if job_number != 0:
                newline = "#BSUB -cwd " + os.getcwd() + "/" + job_name + "\n" + "#BSUB -w " + "done(" + dir_prefix + str(job_number - 1) + ") + "\n"
            else:
                newline = "#BSUB -cwd " + os.getcwd() + "/" + job_name + "\n"
```
Monitoring Jobs and Queues
Why monitor?

• To understand the usage of various queues

• Better decide the queues to submit jobs to

• To observe the progress of a submitted job

• To better schedule several jobs on the same or different queues
Monitor tools

The three most important commands for monitoring the jobs are:

- **bjobs**: Displays and filters information about LSF jobs. Specify one or more job IDs to display information about specific jobs.
  - `% bjobs -pl` Display detailed information of all pending jobs of the invoker.
  - `% bjobs -ps` Display only pending and suspended jobs.
  - `% bjobs -u all -a` Display all jobs of all users.

- **bqueues**: Displays information about queues.
  - `-w` Displays queue information in a wide format. Fields are displayed without truncation.
  - `queue_name ...` Displays information about the specified queues.

- **bhosts**: Displays hosts and their static and dynamic resources.
  - `-l` Displays host information in a (long) multi-line format.
#!/bin/bash

mainpath="/scratch/altman.ro/ToricCY"

esclist=`cat $mainpath/escdirs`

username="altman.ro"

queues=`bjobs -w -r 2>/dev/null | tail -n +2 | sed -r s+/ /g | rev | cut -d' ' -f7 | rev | sort -u`

echo "Running cores on:"
for queue in $queues; do
totmax=`bqueues $queue | tail -n1 | sed -r s+/ /g | cut -d' ' -f5`
totopen=`echo "$(bhosts $(bqueues -l $queue | grep "HOSTS:" | sed 's/.*\(.*\)\(.*\)/\1/g') | grep ok" | sed -r s+/ /g | cut -d' ' -f4,5 | sed 's/\(.*\)\(.*\)/1-2/g' | tr 'n' 'n' | tr 'r' 'r' | bc | (bjobs -s -sum -q $queue | tail -n1 | cut -d' ' -f1)" | bc`
tot=$(($totopen<$totmax?$totopen:$totmax))
runfrac=`echo "$(echo "$bjobs -w -r -q $queue 2>/dev/null | tail -n +2 | sed -r s+/ /g | cut -d' ' -f6 | sed 's/\(.*\)\(.*\)/1-2/g' | tr 'n' 'n' | tr 'r' 'r' | bc | $tot")" | bc`
runpercent=%03d $((100*$runfrac))
done

echo "....$queue:....$runpercent%....$runfrac"

done

pendfrac=`busers -w $username | tail -n +2 | sed -r s+/ /g | cut -d' ' -f5,10 | tr ' ' '/'`
pendpercent=%03d $((100*$pendfrac | bc))

echo "Pending cores:....$pendpercent%....$pendfrac"

done

dirs=`ls -d $mainpath/* 2>/dev/null | rev | cut -d' ' | rev`
for d in $dirs; do
escflag=0
for esc in $esclist; do
  if [ "$d" == "$esc" ]; then
    escflag=1
    break
  fi
done
if [ "$escflag" -eq "0" ]; then
  $mainpath/monitorh11 $d
fi
done

1) Get the list of queues that are being used by your job
2) Get the total number of cores free on all the queues that your job uses.
3) Get the total number of cores on the queues
4) Get percentage of cores yet to be used.
Resources

• LSF Command reference (http://nuweb12.neu.edu/rc/wp-content/uploads/2014/03/lsf_command_ref.pdf)

• Submitting jobs on the Discovery cluster (http://nuweb12.neu.edu/rc/?page_id=18#10g)


• Visit the Research Computing office at 2 Ell Hall (behind the NEU bookstore) for any help/questions

• Check this for office hours (http://nuweb12.neu.edu/rc/?page_id=24)
Questions??
Contact Us

• Contact “researchcomputing@neu.edu” if you have questions or need help at any time

• Research Computing Staff are part of Information Technology Services (ITS) located in 2 Ell Hall, 360 Huntington Avenue, Boston, MA 02115
Thank you
Have a nice day ahead !!
Reference Material
Software Modules
module avail

• To see the list of available modules
module whatis <module_name>

- To see information of the module and its dependencies with other modules

```
[ghosh.s@discovery2 ~]$ module whatis matlab_dce_2013b
matlab_dce_2013b  : loads the modules environment for Matlab 2013b with Distributed Computing Engine (256 seat license) and all ToolBoxes, executable and script files

Needs the following modules to be loaded as prerequisites

module load gnu-4.4-compilers (for certain run-time kernel libraries)
module load fftw-3.3.3
module load platform-mpi
module load oracle_java_1.7u40

Put these module load commands in your .bashrc file that is found in your /home/<user>-id> directory.
```

[ghosh.s@discovery2 ~]$
module list

• To see the list of your loaded modules
• To load modules, use “module load ..” as shown for the particular module
• Best Idea: To add the module load statements to the .bashrc file
• Note: Order of modules loaded is IMPORTANT !!!

[ghosh.s@discovery2 ~]$ module list
Currently Loaded Modulefiles:
    1) gnu-4.4-compilers  3) gnu-4.8.1-compilers  5) oracle_java_1.7u40
    2) platform-mpi       4) fftw-3.3.3            6) hadoop-2.4.1
[ghosh.s@discovery2 ~]$
HANDLING ZOMBIE PROCESSES ON THE DISCOVERY CLUSTER

Research Computing (ITS)
Northeastern University
ZOMBIE PROCESS

• Defunct processes that have not been cleaned up by the system

• Cannot be killed by using kill() system call

• Prevent similar jobs from starting

• Use up system resource

• May even corrupt user shell
HANDLING ZOMBIE PROCESSES

Create the correct submit script

Create the kill-zombies script

Create the kill-user-jobs script
• Needed to find the hosts on which your application ran (and may have created zombies !!)
Kill-Zombies Script

```
#!/bin/bash
IFS="\n"
for (( i=0; i<5; i++ ));
do
  for line in `cat ./host-enumerate`
do
    temp_host="$line"
    ssh -C $temp_host "$PWD/kill_user_jobs" >> kill_details.txt
done
done
#Remember to run "rm host-enumerate" from this folder before running another job from the same folder
```
Kill-Zombies Script

- Iterates over all the different nodes used by your application and kills the zombies created by your application

- Nodes used by the application are obtained from host-enumerate file

- The command `echo "text from fig2" > kill_zombies` quickly creates the kill_zombies script from command line.

- The command `chmod +x kill_zombies` gives it the permission to run

- The command `./kill_zombies` runs the kill_zombies file
Kill-User-Jobs

• Needed to kill all the zombie processes left by the user application.

• Can be created using the command

```bash
#!/bin/bash
killall -u nilay.roy
```

Fig. 3

• The command `chmod +x kill_user_jobs` gives it the permission to run.
Output Expected from your run ...
Research Computing Seminar – Linux Fundamentals and New User Primer for Discovery Cluster Users
Linux fundamentals – commands and navigation
**ls**

- You can list the contents of a directory with `ls`
- List out files and directories in current path

```
paul@peshar:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
paul@peshar:~$
```

**ls -a**

- A frequently used option with `ls` is `-a` to show all files
- Showing all files means including the hidden files
- When a file name on a Unix file system starts with a dot, it is considered a hidden file and it doesn’t show up in regular file listings

```
paul@peshar:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
paul@peshar:~$ ls -a
allfiles.txt  .bash_profile  dmesg.txt  .lesshst  stuff
..  .bash_history  .bashrc  httpd.conf  .sah  summer.txt
paul@peshar:~$
```
**ls -l**

- Many times you will be using options with `ls` to display the contents of the directory in different formats or to display different parts of the directory
- Typing just `ls` gives you a list of files in the directory
- Typing `ls -l` gives you a long listing

```
paul@pasha:~$ ls -l
total 23992
-rw-r--r-- 1 paul paul 24506857 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul 14744 2006-03-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8192 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul  4096 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul   0 2006-03-30 22:45 summer.txt
```
paul@pasha:~$ ls -l -h
total 24M
-rw-r--r-- 1 paul paul 24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul 15K 2006-05-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul 0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -lh
total 24M
-rw-r--r-- 1 paul paul 24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul 15K 2006-05-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul 0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -h -l
total 24M
-rw-r--r-- 1 paul paul 24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul 15K 2006-05-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul 0 2006-03-30 22:45 summer.txt
mkdir

• You can create your own directories with mkdir

• You have to give at least one parameter to mkdir, the name of the new directory to be created

    paul@laika:~$ mkdir MyDir
    paul@laika:~$ cd MyDir
    paul@laika:~/MyDir$ ls -al
    total 0
    drwxr-xr-x 2 paul paul 4096 2007-01-10 21:13 ..
    drwxr-xr-x 2 paul paul 4096 2007-01-10 21:13 ..
    paul@laika:~/MyDir$ mkdir stuff
    paul@laika:~/MyDir$ mkdir otherstuff
    paul@laika:~/MyDir$ ls -l
    total 0
    drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 otherstuff
    drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 stuff
    paul@laika:~/MyDir$

mkdir -p

• When given the option -p, then mkdir will create parent directories as needed

    paul@laika:~$ mkdir -p MyDir2/MySubdir2/ThreeDeep
    paul@laika:~$ ls MyDir2
    MySubdir2
    paul@laika:~$ ls MyDir2/MySubdir2
    ThreeDeep
    paul@laika:~$ ls MyDir2/MySubdir2/ThreeDeep/
**rmdir**

- When a directory is empty, you can use `rmdir` to remove the directory

```
paul@laika:~/mydir$ rmdir otherstuff
paul@laika:~/MyDir$ ls
stuff
paul@laika:~/MyDir$ cd ..
paul@laika:~$ rmdir MyDir
rmdir: MyDir/: Directory not empty
paul@laika:~$ rmdir MyDir/stuff
paul@laika:~$ rmdir MyDir
```

**rmdir -p**

- Similar to the `mkdir -p` option, you can also use `rmdir` to recursively remove directories

```
paul@laika:~$ mkdir -p dir/subdir/subdir2
paul@laika:~$ rmdir -p dir/subdir/subdir2
paul@laika:~$
```
cd

• You can change your current directory with the cd command (Change Directory)

    paul@laika$ cd /etc
    paul@laika$ pwd
    /etc
    paul@laika$ cd /bin
    paul@laika$ pwd
    /bin
    paul@laika$ cd /home/paul/
    paul@laika$ pwd
    /home/paul

cd ~

• Just typing cd without a target directory, will put you in your home directory

• Typing cd ~ has the same effect

    paul@laika$ cd /etc
    paul@laika$ pwd
    /etc
    paul@laika$ cd
    paul@laika$ pwd
    /home/paul
    paul@laika$ cd ~
    paul@laika$ pwd
    /home/paul
cd ..

- To go to the parent directory (the one just above your current directory in the directory tree), type `cd ..`

```bash
paul@laika$ pwd
/home/paul
paul@laika$ cd ..
paul@laika$ pwd
/home/share
```

cd -

- Another useful shortcut with cd is to just type `cd -` to go to the previous directory

```bash
paul@laika$ pwd
/home/paul
paul@laika$ cd /etc
paul@laika$ pwd
/etc
paul@laika$ cd -
/home/paul
paul@laika$ cd -
/etc
```

pwd

- The you are here sign can be displayed with the `pwd` command (Print Working Directory)

```bash
paul@laika:~$ pwd
/home/paul
```
**rm**

- When you no longer need a file, use `rm` to remove it.
- Unlike some graphical user interfaces, the command line in general does not have a waste bin or trash can to recover files.
- When you use `rm` to remove a file, the file is gone.
- Therefore, be careful when removing files!

```
paul@laika:~/test$ ls
BigBattle  SinkoDeMayo
paul@laika:~/test$ rm BigBattle
paul@laika:~/test$ ls
SinkoDeMayo
```

**rm -i**

- To prevent yourself from accidentally removing a file, you can type `rm -i`.

```
paul@laika:~/Linux$ touch brel.txt
paul@laika:~/Linux$ rm -i brel.txt
rm: remove regular empty file `brel.txt'? y
paul@laika:~/Linux$
```
rm -rf

- By default, rm -r will not remove non-empty directories
- However rm accepts several options that will allow you to remove any directory
- The rm -rf statement will erase anything (providing that you have the permissions to do so)
- When you are logged on as root, be very careful with rm -rf (the f means force and the r means recursive) since being root implies that permissions don’t apply to you
- You can literally erase your entire file system by accident

```
paul@laika:~$ ls test
SlinkeDeMayo
paul@laika:~$ rm test
rm: cannot remove 'test': Is a directory
paul@laika:~$ rm -rf test
paul@laika:~$ ls test
ls: test: No such file or directory
```

cp

- To copy a file, use cp with a source and a target argument
- If the target is a directory, then the source files are copied to the target directory
To copy complete directories, use `cp -r` (the `-r` option forces recursive copying of all files in all subdirectories).

You can also use `cp` to copy multiple files into a directory.

In this case, the last argument must be a directory.
To prevent `cp` from overwriting existing files, use the `-i` (for interactive) option

```
paul@laika:~/test$ cp fire water
paul@laika:~/test$ cp -i fire water
cp: overwrite 'water'? no
paul@laika:~/test$
```

To preserve permissions and time stamps from source files, use `cp` `-p`

```
paul@laika:~/perms$ cp file* cp
paul@laika:~/perms$ cp -p file* cpp
paul@laika:~/perms$ ll *
-rwx------ 1 paul paul 0 2008-08-25 13:26 file33
-rwx------ 1 paul paul 0 2008-08-25 13:26 file42

cp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:34 file33
-rwx------ 1 paul paul 0 2008-08-25 13:34 file42
cpp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:26 file33
-rwx------ 1 paul paul 0 2008-08-25 13:26 file42
```
mv

- Use mv to rename a file or to move the file to another directory
- When you need to rename only one file then mv is the preferred command to use

```
paul@laika:~/test$ touch file100
paul@laika:~/test$ ls
file100
paul@laika:~/test$ mv file100 ABC.txt
paul@laika:~/test$ ls
ABC.txt
paul@laika:~/test$ 
```

file ownership

user owner and group owner

- The users and groups of a system can be locally managed in `/etc/passwd` and `/etc/group`, or they can be in a NIS, LDAP, or Samba domain
- These users and groups can own files
- Actually, every file has a user owner and a group owner

```
paul@RHELv4u4:~/test$ ls -l
total 24
-rw-r--r-- 1 paul paul 17 Feb  7 11:53 file1
-rw-r--r-- 1 paul paul 106 Feb  5 17:04 file2
-rw-r--r-- 1 paul proj 984 Feb  5 15:38 data.odt
-rw-r--r-- 1 root root 0 Feb  7 16:07 stuff.txt
paul@RHELv4u4:~/test$ 
```
• User paul owns three files, two of those are also owned by the group paul
• data.odt is owned by the group proj
• The root user owns the file stuff.txt, as does the group root

chgrp
• You can change the group ownership of a file using the chgrp command

```
root@laika:/home/paul# touch FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root root 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chgrp paul FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
```

chown
• The user owner of a file can be changed with chown command

```
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root paul 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chown paul FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
```
• You can also use chown to change both the user owner and the group owner

```
root@laika:/home/paul# ls -l FileForPaul
-rwxr-xr-x 1 paul paul 0 2009-08-06 14:11 FileForPaul
root@laika:/home/paul# chown root:project42 FileForPaul
-rw-r--r-- 1 root project42 0 2009-08-06 14:11 FileForPaul
```

list of special files

• When you use ls -l, for each file you can see ten characters before the user and group owner
• The first character tells us the type of file
• Regular files get a -, directories get a d, symbolic links are shown with an l, pipes get a p, character devices a c, block devices a b, and sockets as s

<table>
<thead>
<tr>
<th>first character</th>
<th>file type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>normal file</td>
</tr>
<tr>
<td>d</td>
<td>directory</td>
</tr>
<tr>
<td>l</td>
<td>symbolic link</td>
</tr>
<tr>
<td>p</td>
<td>named pipe</td>
</tr>
<tr>
<td>b</td>
<td>block device</td>
</tr>
<tr>
<td>c</td>
<td>character device</td>
</tr>
<tr>
<td>s</td>
<td>socket</td>
</tr>
</tbody>
</table>
• Below a screenshot of a character device (the console) and a block device (the hard disk)

```
paul@debian6lt-$ ls -1d /dev/console /dev/sda
  crw------- 1 root root 5, 1 Mar 15 12:45 /dev/console
  brw-rw--- 1 root disk 8, 0 Mar 15 12:45 /dev/sda
```

• Here you can see a directory, a regular file and a symbolic link

```
paul@debian6lt-$ ls -1d /etc /etc/hosts /etc/motd
drwxr-xr-x 128 root root 12288 Mar 15 18:34 /etc
-rw-r--r-- 1 root root 372 Dec 10 17:36 /etc/hosts
lrwxrwxrwx 1 root root 15 Dec 5 10:36 /etc/motd -> /var/run/motd
```

**permissions**

**rwx**

• The nine characters following the file type denote the permissions in three triplets

• A permission can be r for read access, w for write access, and x for execute

• You need the r permission to list (ls) the contents of a directory

• You need the x permission to enter (cd) a directory

• You need the w permission to create files in or remove files from a directory
three sets of rwx

- We already know that the output of `ls -l` starts with ten characters for each file.
- This screenshot shows a regular file (because the first character is a -)

```
paul@RHELv4u4:~$ ls -l proc42.bash
-rwxr-xr-x 1 paul proj 984 Feb 6 12:01 proc42.bash
```

- Below is a table describing the function of all ten characters.
When you are the user owner of a file, then the user owner permissions apply to you.

The rest of the permissions have no influence on your access to the file.

When you belong to the group that is the group owner of a file, then the group owner permissions apply to you.

The rest of the permissions have no influence on your access to the file.

When you are not the user owner of a file and you do not belong to the group owner, then the others permissions apply to you.

The rest of the permissions have no influence on your access to the file.

---

**Table 25.3. Unix file permissions position**

<table>
<thead>
<tr>
<th>position</th>
<th>characters</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>this is a regular file</td>
</tr>
<tr>
<td>2-4</td>
<td>rwx</td>
<td>permissions for the user owner</td>
</tr>
<tr>
<td>5-7</td>
<td>r-x</td>
<td>permissions for the group owner</td>
</tr>
<tr>
<td>8-10</td>
<td>r--</td>
<td>permissions for others</td>
</tr>
</tbody>
</table>
permission examples

• Some example combinations on files and directories are shown in this screenshot

• The name of the file explains the permissions

```
paul@laika:~$/perm$ ls -lh
total 12K
-dw-r-xr-x 2 paul paul 4.0K 2007-02-07 22:26 AllEnter_UserCreateDelete
-rwx-rwxwx 1 paul paul 0 2007-02-07 22:21 EveryoneFullControl.txt
-r-------- 1 paul paul 0 2007-02-07 22:21 OnlyOwnersRead.txt
-rwxrwx----- 1 paul paul 0 2007-02-07 22:21 OwnersAll_ReadNothing.txt
-dr-xr-x--- 2 paul paul 4.0K 2007-02-07 22:25 UserAndGroupEnter
-dr-------- 2 paul paul 4.0K 2007-02-07 22:25 OnlyUserEnter
paul@laika:~/perm$
```

• To summarize, the first rwx triplet represents the permissions for the user owner

• The second triplet corresponds to the group owner; it specifies permissions for all members of that group

• The third triplet defines permissions for all other users that are not the user owner and are not a member of the group owner
setting permissions (chmod)

- Permissions can be changed with chmod
- The first example gives the user owner execute permissions

```
paul@laika:~/perms$ ls -l permissions.txt
-rw-r--r--  1 paul paul  0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod u+x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxr--r--  1 paul paul  0 2007-02-07 22:34 permissions.txt
```

- This example removes the group owners read permission

```
paul@laika:~/perms$ chmod g-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx---r--  1 paul paul  0 2007-02-07 22:34 permissions.txt
```

- This example removes the others read permission

```
paul@laika:~/perms$ chmod o-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx------  1 paul paul  0 2007-02-07 22:34 permissions.txt
```

- This example gives all of them the write permission

```
paul@laika:~/perms$ chmod a+w permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-w--w-  1 paul paul  0 2007-02-07 22:34 permissions.txt
```
• You don’t even have to type the a

```
paul@laika:~/perms$ chmod +x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

• You can also set explicit permissions

```
paul@laika:~/perms$ chmod u=rw permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw--wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

• Feel free to make any kind of combination

```
paul@laika:~/perms$ chmod u=rw,g=rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

• Even fishy combinations are accepted by chmod

```
paul@laika:~/perms$ chmod u=rw,u+g=rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxrw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```
setting octal permissions

- Most Unix administrators will use the old school octal system to talk about and set permissions.
- Look at the triplet bitwise, equating r to 4, w to 2, and x to 1.

<table>
<thead>
<tr>
<th>binary</th>
<th>octal</th>
<th>permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>---</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>--x</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>-w-</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>-wx</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>r--</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>r-x</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>rw-</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>rwx</td>
</tr>
</tbody>
</table>

- This makes 777 equal to rwxrwxrwx and by the same logic, 654 means rw-r-xr--.
- The chmod command will accept these numbers.
mkdir -m

- When creating directories with mkdir you can use the -m option to set the mode
- This screenshot explains
**hard links**

**creating hard links**

- When we create a hard link to a file with `ln`, an extra entry is added in the directory
- A new file name is mapped to an existing inode

```
paul@RHELx4u4:~/test$ ln file2 hardlink_to_file2
paul@RHELx4u4:~/test$ ls -ll
```

```
total 24
817266 -rw-r--r-- 1 paul paul 0 Feb 5 15:38 file1
817270 -rw-r--r-- 2 paul paul 92 Feb 5 15:42 file2
817268 -rw-r--r-- 1 paul paul 0 Feb 5 15:38 file3
817270 -rw-r--r-- 2 paul paul 92 Feb 5 15:42 hardlink_to_file2
paul@RHELx4u4:~/test$
```

- Both files have the same inode, so they will always have the same permissions and the same owner
- Both files will have the same content
- Actually, both files are equal now, meaning you can safely remove the original file, the hardlinked file will remain
- The inode contains a counter, counting the number of hard links to itself
- When the counter drops to zero, then the inode is emptied
inodes

• An inode is a data structure that contains metadata about a file

• When the file system stores a new file on the hard disk, it stores not only the contents (data) of the file, but also extra properties like the name of the file, the creation date, its permissions, the owner of the file, and more

• All this information (except the name of the file and the contents of the file) is stored in the inode of the file

• The ls -l command will display some of the inode contents, as seen in this screenshot

```
root@rhel53 ~ $ ls -ld /home/project42/
drwxr-xr-x 4 root proj 4.0K Mar 27 14:29 /home/project42/
```

• Each inode has a unique number (the inode number)

• You can see the inode numbers with the ls -li command

```
paul@RHEL4u4:/test$ touch file1
paul@RHEL4u4:/test$ touch file2
paul@RHEL4u4:/test$ touch file3
paul@RHEL4u4:/test$ ls -li
total 12
81726E -rw-r--r-- 1 paul paul 0 Feb 5 15:38 file1
817267 -rw-r--r-- 1 paul paul 0 Feb 5 15:38 file2
817268 -rw-r--r-- 1 paul paul 0 Feb 5 15:38 file3
paul@RHEL4u4:/test$
```

• These three files were created one after the other and got three different inodes (the first column)
• All the information you see with this `ls` command resides in the inode, except for the filename (which is contained in the directory)

**inode and file contents**

• Lets put some data in one of the files

```
paul@RHELv4u4:~$ ls -li
total 16
 817266 -rw-rw-r-- 1 paul paul 0 Feb 5 15:38 file1
 817270 -rw-rw-r-- 1 paul paul 0 Feb 5 15:42 file2
 817268 -rw-rw-r-- 1 paul paul 0 Feb 5 15:39 file3
paul@RHELv4u4:~$ cat file2
It is winter now and it is very cold. We do not like the cold, we prefer hot summer nights.
paul@RHELv4u4:~$```

• The data that is displayed by the `cat` command is not in the inode, but somewhere else on the disk

• The inode contains a pointer to that data
finding hard links

- You can use the `find` command to look for files with a certain inode
- The screenshot below shows how to search for all filenames that point to inode 817270
- Remember that an inode number is unique to its partition

```
paul@RHELv4u4:-/test$ find / -iname "817270" 2> /dev/null
/home/paul/test/file2
/home/paul/test/hardlink_to_file2
```

symbolic links

- Symbolic links (sometimes called soft links) do not link to inodes, but create a name to name mapping
- Symbolic links are created with `ln -s`
- As you can see below, the symbolic link gets an inode of its own

```
paul@RHELv4u4:-/test$ ln -s file2 symlink_to_file2
paul@RHELv4u4:-/test$ ls -li
total 32
817270 -rw-rw-r-- 1 paul paul 13 Feb 5 17:06 file1
817270 -rw-rw-r-- 2 paul paul 106 Feb 5 17:04 file2
817268 -rw-rw-r-- 1 paul paul 0 Feb 5 15:30 file3
817270 -rw-rw-r-- 2 paul paul 106 Feb 5 17:04 hardlink_to_file2
817276 1wxrwxrwx 1 paul paul 5 Feb 5 16:55 symlink_to_file2 -> file2
paul@RHELv4u4:-/test$
```

- Permissions on a symbolic link have no meaning, since the permissions of the target apply
- Hard links are limited to their own partition (because they point to an inode), symbolic links can link anywhere
removing links
• Links can be removed with `rm`

```
paul@laika:~$ touch data.txt
paul@laika:~$ ln -s data.txt sl_data.txt
paul@laika:~$ ln data.txt hl_data.txt
paul@laika:~$ rm sl_data.txt
paul@laika:~$ rm hl_data.txt
```

find
• The find command can be very useful at the start of a pipe to search for files
• Here are some examples
• Find all files of the entire system and put the list in allfiles.txt

```
find / > allfiles.txt
```
• Find all files in `/etc` and put the list in etcfiles.txt

```
find /etc > etcfiles.txt
```
• Find files that end in `.conf` in the current directory (and all subdirs)

```
find . -name "*.conf"
```
• Find files of type file that end in `.conf`

```
find . -type f -name "*.conf"
```
• Find files of type directory that end in .bak

    find /data -type d -name "*.bak"

gzip - gunzip

• Users never have enough disk space, so compression comes in handy

• The gzip command can make files take up less space

    paul@rhe155 ~$ ls -lh text.txt
    -rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt
    paul@rhe155 ~$ gzip text.txt
    paul@rhe155 ~$ ls -lh text.txt.gz
    -rw-rw-r-- 1 paul paul 760K Apr 17 13:11 text.txt.gz

• You can get the original back with gunzip

    paul@rhe155 ~$ gunzip text.txt.gz
    paul@rhe155 ~$ ls -lh text.txt
    -rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt

tar

• The tar command is used for backing up a directory
**cat**

- The cat command is one of the most universal tools
- All it does is copy standard input to standard output
- You can use cat to display a file on the screen
- If the file is longer than the screen, it will scroll to the end

```bash
paul@laika:~$ cat /etc/resolv.conf
nameserver 194.7.1.4
paul@laika:~$
```

**concatenate**

- cat is short for concatenate
- One of the basic uses of cat is to concatenate files into a bigger file

```bash
paul@laika:~$ echo one > part1
paul@laika:~$ echo two > part2
paul@laika:~$ echo three > part3
paul@laika:~$ cat part1 part2 part3
one
two
three
```

```bash
paul@laika:~$
```
create files

- You can use `cat` to create flat text files
- Type the `cat > winter.txt` command as shown in the screenshot below
- Then type one or more lines, finishing each line with the enter key
- After the last line, type and hold Control (Ctrl) key and press `d`

```
paul@laika:~/test$ cat > winter.txt
It is very cold today!
paul@laika:~/test$ cat winter.txt
It is very cold today!
paul@laika:~/test$
```

- The Ctrl d key combination will send an EOF (End of File) to the running process ending the cat command
Introduction to vi

• The vi editor is installed on almost every Unix
• Linux will very often install vim (vi improved) which is similar

command mode and insert mode

• The vi editor starts in command mode
• In command mode, you can type commands
• Some commands will bring you into insert mode
• In insert mode, you can type text
• The escape key will return you to command mode

Table 18.1. getting to command mode

<table>
<thead>
<tr>
<th>key</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Esc</td>
<td>set vi(m) in command mode.</td>
</tr>
</tbody>
</table>
start typing

- The difference between a A i I o and O is the location where you can start typing
- a will append after the current character and A will append at the end of the line
- i will insert before the current character and I will insert at the beginning of the line
- o will put you in a new line after the current line and O will put you in a new line before the current line

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>start typing after the current character</td>
</tr>
<tr>
<td>A</td>
<td>start typing at the end of the current line</td>
</tr>
<tr>
<td>i</td>
<td>start typing before the current character</td>
</tr>
<tr>
<td>I</td>
<td>start typing at the start of the current line</td>
</tr>
<tr>
<td>o</td>
<td>start typing on a new line after the current line</td>
</tr>
<tr>
<td>O</td>
<td>start typing on a new line before the current line</td>
</tr>
</tbody>
</table>
replace and delete a character

- When in command mode (it doesn't hurt to hit the escape key more than once) you can use the x key to delete the current character
- The big X key (or shift x) will delete the character left of the cursor
- Also when in command mode, you can use the r key to replace one single character
- The r key will bring you in insert mode for just one key press, and will return you immediately to command mode

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>delete the character below the cursor</td>
</tr>
<tr>
<td>X</td>
<td>delete the character before the cursor</td>
</tr>
<tr>
<td>r</td>
<td>replace the character below the cursor</td>
</tr>
<tr>
<td>p</td>
<td>paste after the cursor (here the last deleted character)</td>
</tr>
<tr>
<td>xp</td>
<td>switch two characters</td>
</tr>
</tbody>
</table>


**cut, copy and paste a line**

- When in command mode, `dd` will cut the current line
- `yy` will copy the current line
- You can paste the last copied or cut line after (p) or before (P) the current line

**Table 18.5. cut, copy and paste a line**

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd</td>
<td>cut the current line</td>
</tr>
<tr>
<td>yy</td>
<td>(yank yank) copy the current line</td>
</tr>
<tr>
<td>p</td>
<td>paste after the current line</td>
</tr>
<tr>
<td>P</td>
<td>paste before the current line</td>
</tr>
</tbody>
</table>

**cut, copy and paste lines**

- When in command mode, before typing `dd` or `yy`, you can type a number to repeat the command a number of times
- Thus, `5dd` will cut 5 lines and `4yy` will copy (yank) 4 lines
- That last one will be noted by vi in the bottom left corner as "4 line yanked"
When in command mode, the 0 and the caret ^ will bring you to the start of the current line, whereas the $ will put the cursor at the end of the current line.

You can add 0 and $ to the d command, d0 will delete every character between the current character and the start of the line.

Likewise d$ will delete everything from the current character till the end of the line.

Similarly y0 and y$ will yank till start and end of the current line.
**join two lines and more**

- When in command mode, pressing J will append the next line to the current line
- With yyp you duplicate a line and with ddp you switch two lines

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>join two lines</td>
</tr>
<tr>
<td>yyp</td>
<td>duplicate a line</td>
</tr>
<tr>
<td>ddp</td>
<td>switch two lines</td>
</tr>
</tbody>
</table>

**words**

- When in command mode, w will jump to the next word and b will move to the previous word
- w and b can also be combined with d and y to copy and cut words (dw db yw yb)

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>forward one word</td>
</tr>
<tr>
<td>b</td>
<td>back one word</td>
</tr>
<tr>
<td>3w</td>
<td>forward three words</td>
</tr>
<tr>
<td>dw</td>
<td>delete one word</td>
</tr>
<tr>
<td>yw</td>
<td>yank (copy) one word</td>
</tr>
<tr>
<td>5yb</td>
<td>yank five words back</td>
</tr>
<tr>
<td>7dw</td>
<td>delete seven words</td>
</tr>
</tbody>
</table>
save (or not) and exit

- Pressing the colon : will allow you to give instructions to vi (technically speaking, typing the colon will open the ex editor)
- :w will write (save) the file, :q will quit an unchanged file without saving, and :q! will quit vi discarding any changes
- :wq will save and quit and is the same as typing ZZ in command mode

<table>
<thead>
<tr>
<th>command</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>:w</td>
<td>save (write)</td>
</tr>
<tr>
<td>:w name</td>
<td>save as name</td>
</tr>
<tr>
<td>:q</td>
<td>quit</td>
</tr>
<tr>
<td>:wq</td>
<td>save and quit</td>
</tr>
<tr>
<td>ZZ</td>
<td>save and quit</td>
</tr>
<tr>
<td>:q!</td>
<td>quit (discarding your changes)</td>
</tr>
<tr>
<td>:w!</td>
<td>save (and write to non-writable file!)</td>
</tr>
</tbody>
</table>

- With :w! vi will try to chmod the file to get write permission (this works when you are the owner) and will chmod it back when the write succeeds
- This should always work when you are root (and the file system is writable)
Connections and File Transfers
Connecting to Discovery Cluster

- [http://nuweb12.neu.edu/rc/?page_id=75](http://nuweb12.neu.edu/rc/?page_id=75)

- The method to connect to the Discovery cluster is via ssh (secure shell)

- ssh -X or ssh -Y ensures X11 forwarding is enabled and you can launch windows/GUI’s from Discovery login nodes

- An example login can be shown using your cluster account credentials which is your “myneu” username and password

- Discovery cluster login authentication is done via Northeastern University Active Directory and you will only be able to login after your account application is approved by ITS Research Computing and the Research Computing Committee
• To login from Windows clients directly we recommend using Putty for ssh and Xming for X11 forwarding


• Please note this is an executable and can be run directly


• This is an installation package and must be installed selecting all the default options

Connecting to Discovery Cluster using ssh with X11 forwarding from Windows Clients

Assuming you have Putty and Xming installed follow the steps shown below:

1) Start Xming:
2) If you see any message from your firewall or antivirus select allow and the Xming icon will appear on the task bar. If you hover your mouse over it you see: “Xming Server:0.0”.

3) Start Putty:
4) For Host Name enter discovery.neu.edu and the word Discovery in the Saved Session test box as shown in 3) above.
5) Next select SSH and X11 from the Category pane. This is shown below and make sure Enable X11 forwarding is checked.
6) Go back to the Session in Category pane and select Save so that the profile is saved. You will see Discovery added to the Saved Sessions below Default Setting.
7) Now exit Putty and restart it. Select Discovery and click Load. Then click Open. You will get the login shell as shown below. Accept any certificate that is sent. Then login in with your user name and password. You will be logged into one of Discovery login nodes on a round-robin basis.
8) Test X11 forwarding works by starting a GUI like gedit for example as shown below:
9) You may launch as many instances of Putty. Each one will give you one terminal session.

10) When done logout from the Discovery login and your connection will exit.

11) Finally shut Xming down by right clicking on the Xming icon on the task bar on your Windows client and selecting Exit. You will be asked to confirm exit.
• On Windows Desktops rather than using a windows ssh client like Putty and XMing (for X11 windows forwarding) it is recommended that Linux be installed using Oracle VM Virtual Box

• To transfer files and folders to and from the cluster use either “sFtp” or “rsync”

• “rsync” is the preferred method as it uses secure encoding and can restart if connection is broken

• You can also use a free sFtp Windows or Linux Utility like FileZilla

• The port to use is 22

• It can be downloaded from https://filezilla-project.org/ – ensure you get the “FileZilla Client” for install

• For Host use “discovery.neu.edu”, for Username and Password use your myneu credentials

• Then click “Quickconnect”

• If you get a warning check the box to update keys and proceed
• If you have issues using “discovery.neu.edu” when you ssh into the cluster you can directly opt for a login node

• In this case replace “discovery.neu.edu” with “discovery2.neu.edu” or “discovery4.neu.edu” that will take you directly to a login node without using round-robin DNS

• If you see warning messages like “DNS SPOOFING DETECTED” or “REMOTE HOST IDENTIFICATION HAS CHANGED” do one of the following:

  (a) Delete the “known_hosts” file in your “.ssh” folder in the home directory of the machine from which you are logging into the Discovery Cluster

  (b) If you do not want to do the above every time you see the warning message and want something more permanent then do the following:
• Delete the “known_hosts” file in your “.ssh” folder in the home directory of the machine from which you are logging into the Discovery Cluster

• Login first to “discovery2.neu.edu” and logout. Then login to “discovery4.neu.edu” and log out

• Now in your “known_hosts” file you will see two entries similar to what is as shown below:

discovery2.neu.edu,129.10.0.22 ssh-rsa m4KOOPWpxnTUqLdqAy7PCfG+FDZADOwNSAlkJ251CCv21V6fI9I6FUfh/ZwYEivLmLFbXLacLwU4Cgv5fiEyAQlSsqYyw==
discovery4.neu.edu,129.10.0.24 ssh-rsa /W4XdcZj1cmJQAx2qznj3JvAz+87gM97EdzGtZe3egDoEkoYACwUzdApCUIF8oivNjtRbEkqyYCElevPWMy6g+1dy6ow==

• Copy both and replace “discovery2.neu.edu,129.10.0.22 ssh-rsa” and “discovery4.neu.edu,129.10.0.24 ssh-rsa” with “discovery.neu.edu ssh-rsa”

• Remove the ip address in each case

• The file will now look like:

discovery2.neu.edu,129.10.0.22 ssh-rsa m4KOOPWpxnTUqLdqAy7PCfG+FDZADOwNSAlkJ251CCv21V6fI9I6FUfh/ZwYEivLmLFbXLacLwU4Cgv5fiEyAQlSsqYyw==
discovery4.neu.edu,129.10.0.24 ssh-rsa /W4XdcZj1cmJQAx2qznj3JvAz+87gM97EdzGtZe3egDoEkoYACwUzdApCUIF8oivNjtRbEkqyYCElevPWMy6g+1dy6ow==
discovery.neu.edu ssh-rsa +m4KOOPWpxnTUqLdqAy7PCfG+FDZADOwNSAlkJ251CCv21V6fI9I6FUfh/ZwYEivLmLFbXLacLwU4Cgv5fiEyAQlSsqYyw==
discovery.neu.edu ssh-rsa /W4XdcZj1cmJQAx2qznj3JvAz+87gM97EdzGtZe3egDoEkoYACwUzdApCUIF8oivNjtRbEkqyYCElevPWMy6g+1dy6ow==

• Save the “known_hosts” file

• Now when you ssh using “discovery.neu.edu” you will never see the warning messages