

mrunc tutorial

- on GNU/Linux open up a terminal and type:

```
$ hostname  
$ date
```

- what do these programs do?



- mrun (**m**ultiple **r**un) is a program which allows you to run a program on multiple machines
- there is some [documentation](http://floppsie.comp.glam.ac.uk/csn/csn.html) (`http://floppsie.comp.glam.ac.uk/csn/csn.html`) under sections 8, 9 and 10
- it is hoped that this tutorial will also bootstrap your knowledge

Tiny example

- suppose we have a program `hostname` which we want to run on two different machines in parallel
 - we could use the command line program `ssh` to achieve this end, but it involves much typing and after the `n`th time of running, becomes tedious
- we can use `mrunc` instead, but we firstly need to create a `par` file
- in our tiny example we will call this filename `hostname.par`
- create a file called `hostname.par` using `gedit`

Contents of hostname.par

```
#  
# example par file to run hostname on two machines  
#  
par  
    processor 0 (localhost) [::] hostname ;  
    processor 1 (localhost) [::] hostname ;  
end  
  
timeout 2h ;  
# terminal 0 1 ;
```

■ now open up a terminal and type

Contents of hostname.par

```
$ mrun -f hostname.par  
Password:  
press the <enter> key to terminate  
<processor 0>:  
<processor 0>:fred@j210-03:$ hostname  
<processor 0>:j210-03  
<processor 0>:fred@j210-03:$  
<processor 1>:  
<processor 1>:fred@j210-03:$ hostname  
<processor 1>:j210-04  
<processor 1>:fred@j210-03:$  
  
halting and tidying up.. done
```

Contents of `hostname.par`

- we notice that
 - `mrun` will prompt us for a password, you need to enter your GNU/Linux password here
 - you need to press the enter key to terminate `mrun`
 - `mrun` randomly chooses any machine which is available from the chosen pool

- we stipulated we wanted any localhost processor by the field `(localhost)`
 - try changing this to `(x86_64)`
 - does it still work?

Contents of `hostname.par`

- the field `timeout 2h` says to stop running after 2 hours and could be replaced by `timeout 5m` if appropriate
- comments in the `par` file are the `#` character, anything to the right of this is ignored

Contents of hostname.par

- try uncommenting the last line, ie remove the ' #' on the last line
- run the program again

- ```
$ mrun -f hostname.par
```

- what happens?
- now change the par file to execute the program date and run mrun again



## Using the for statement in a par file

- now create a new file `hostname2.par`

## Contents of hostname2.par

```
par
 processor 0 (localhost) [::] hostname ;
 for i in 1 to 6 do
 processor ({i}) (localhost) [::] hostname ;
 end
end

timeout 2h ;
terminal 0 3 4 ;
```

## Contents of hostname2.par

- run this via:

```
$ mrun -f hostname2.par
```

## Contents of hostname2.par

- notice that the `{i}` expands to the value of `i` in the for loop
- now change the contents of the file to

```
par
 processor 0 (localhost) [::] echo 0 ;
 for i in 1 to 6 do
 processor ({i}) (localhost) [::] echo {i} ;
 end
end

timeout 2h ;
terminal 0 3 4 ;
```

- the program `echo` just prints to the console
- run this new `par` file



## Contents of hostname2.par

- try running `mr`
- ```
$ mr --help
```
- find out what all the other options do, hint read the online documentation mentioned at the top of this tutorial