- You may need to create Linux processes that will need to run for hours or days ("long-running processes" or LRPs), and must continue to run on lab machines after you log out. Staying logged in for several hours and locking your screen is a common (but bad!) way of achieving this.
- This causes inconvenience to other users, and specifically breaks the DoC screen locking policy - which was requested by students. This policy is:

  > *No-one should lock any lab computer for more than 30 minutes without discussing it with CSG first. The only exception to this is if we have specifically allocated you a project machine.*

- So, here's our advice about how to complete your work in the least disruptive way possible:

- If your problem can be parallelised into one algorithm operating independently on different sets of data, we recommend the **Condor batch processing** framework, which handles the donkey work (starting jobs on idle machines, killing processes off when an interactive user returns, and restarting that job somewhere else). See our local Condor documentation for more information.
- If Condor is not appropriate, you need to find one or more suitable machines most appropriate to run your LRPs on:
  - Shell servers are NEVER the right answer.
  - The batch servers batch1.doc.ic.ac.uk (aka tui) and batch2.doc.ic.ac.uk (aka potoo01) are powerful machines and may well be suitable - but there are only two such machines!
  - Often a Linux lab machine is the best choice, given that lab machines have as many cores as servers nowadays, and desktop processors may run faster than server processors.

- If your program only has a GUI, this is an obstacle to automation and running it while not logged on.
- Can you build a non-interactive, non-graphical version of the software, or use another piece of software instead?
- If it is not possible to eliminate the GUI, you may have no choice but to use a lab machine for long periods of time.
- In this case, please discuss this with us before doing it - please email this info to help@doc.ic.ac.uk telling us (at least):
  - what you're trying to do?
  - how long you believe it will take?
  - why you really truly can't get rid of the GUI?
  - which machine(s) you're thinking of using?
- Perhaps we'll be able to suggest a way of achieving the same goal without using software with a GUI, or suggest you use a non-lab PC.

- Given a non-interactive program, and a chosen machine (let's say matrix15), estimate how long the program should take to run (eg. by trying smaller test cases).
- Also ensure your LRP is not going to inconvenience other users of the same machine (eg. by using all the RAM, CPU power or network bandwidth of the machine). This is very important and is your responsibility.
- Now, work out how to run the program automatically [this is also necessary for using condor]: work out a single command that will run the program in an automatic fashion with the correct data inputs, and capture the output.
  To achieve this, you may need to use a mixture of:
  - commmand line arguments,
  - input redirection - `CMD < INPUTFILE`, or `CMD < /dev/null`,
  - output redirection - `CMD > OUTPUTFILE 2> ERRORFILE`, or `CMD > OUTPUTFILE 2>&1` to redirect stderr (fd 2) to OUTPUTFILE.

  (above in Bourne shell "sh" syntax).

- Wrap the above "run it automatically" command up in a short shell script, called RUNME for instance, roughly of the form:

```
#/!/bin/sh -
cd STARTING_DIRECTORY
CMD ARGS < INPUTFILE > OUTPUTFILE 2> ERRORFILE
```

- Make it executable: `chmod +x RUNME` and then run it from another directory (eg /) using it's absolute path:

```
set dir=‘pwd‘
cd /
$dir/RUNME
```

- Check that it works completely, check the output and error files.
- You can pass RUNME arguments through to your chosen command by adding `$*` (shell command line arguments) to the `CMD ARGS` invocation.

- To run your RUNME script on your chosen machine in the background: do the following (tcsh syntax):

```
nice nohup $dir/RUNME </dev/null >&/dev/null &
```

  That means:

  - Run the script `RUNME` found in directory `$dir`.
  - Detached from keyboard (`</dev/null`)
  - Detached from screen [stdout and stderr] (`>&/dev/null`)
  - In the background (`&`)
  - Make it immune to hangup signals (`nohup`)
  - reduce it's scheduling priority (be `nice`) to favour other processes
- Now, log out and back in again. Check that RUNME is still running by: `ps auxww|grep RUNME`
- Monitor your LRP for a while with top to check it's resource utilization will not overwhelm the machine.
- Check the output file periodically (NB: the most recent 4K of output may not appear in the output until the output buffer is full, or the file is flushed).

- Check that the LRP finishes in a reasonable amount of time - kill it off if it fails to terminate in twice the estimated time! Do not just leave it running for days when it should have finished in 2 hours!
- If you follow these guidelines, you should be able to get your long-running work done, without inconveniencing other users.
- **Don't Forgot to Optimize the Hotspots**: Orthogonal to the above, before running a program for days, try to optimize it's performance on a relatively small dataset:
- **Profile** your program's runtime behaviour, to see where it's really spending the time. No programmer fully understands the runtime behaviour of their own code - profiling always surprises you! Then optimize the **hotspot** - the small part of the program that is taking the most amount of time!
- Some members of CSG have a lot of expertise in this area, it's well worth discussing such problems with us.