Problem 1:

mlockall() locks all pages that have been mapped to the address space of the calling process to the RAM including code segment, data segment, stack segment, shared libraries, user space kernel data, shared memory and memory mapped files.

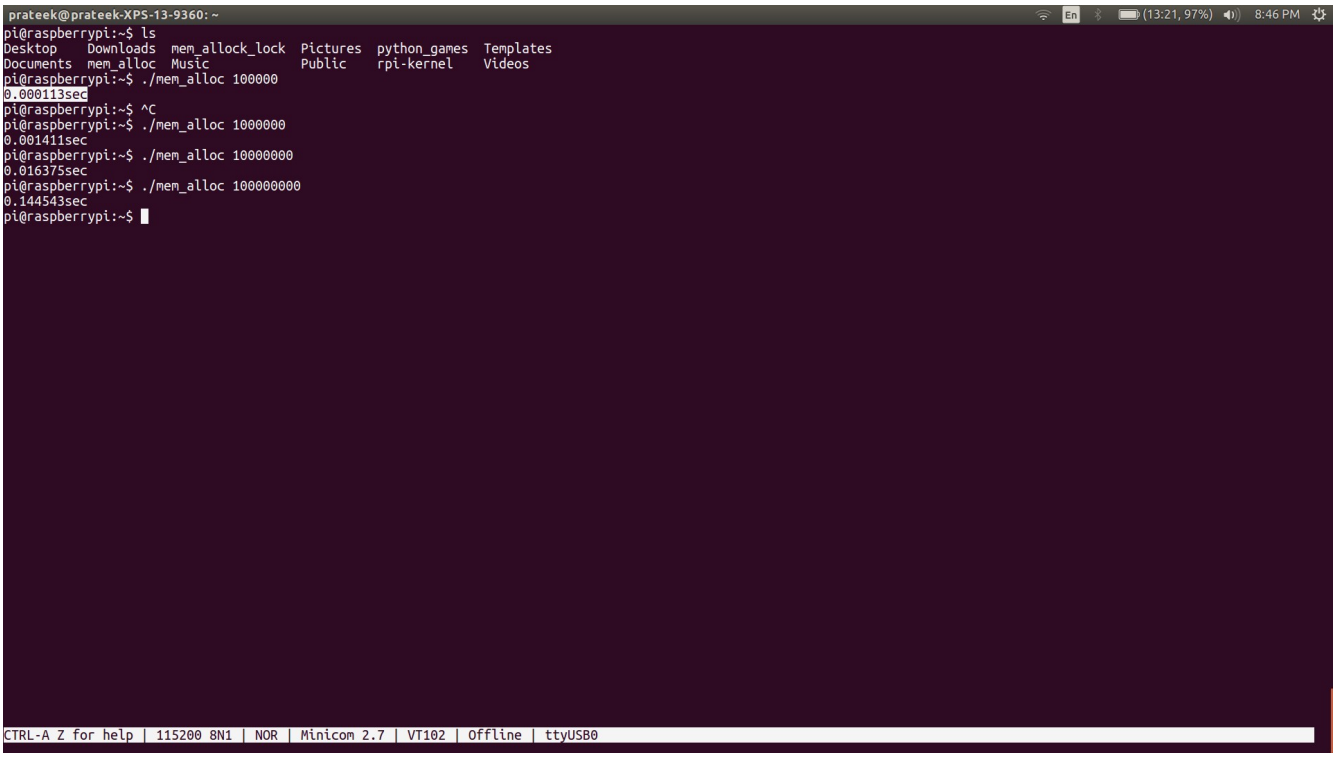
The MCL_CURRENT argument locks all pages which are mapped to the address space of the process.

The MCL_FUTURE locks all pages which will become mapped to the address space of the process in the future.

Threads started after a call to mlockall(MCL_CURRENT | MCL_FUTURE) will generate page faults immediately since the new stack is immediately forced to RAM (due to the MCL_FUTURE flag).

The **flags** argument is constructed by bitwise OR of the above arguments.

Problem 2:



100KB: 0.000113 sec

1MB: 0.001411 sec

10MB: .016375 sec

100MB: 0.144543 sec

Problem 3:

Since modification in the kernel is beyond the scope of the problem, certain practices while writing code may be used to prevent unpredictable delays in memory access, namely:

When dealing with large data structures, the use of spatial and temporal locality may be a good way to go. Procedures such as cache blocking may be used to improve spatial locality and reduce memory access time especially in BLAS operations involving matrices. Reusing variables and utilizing registers, may also be a good way to reduce memory access time. Knowledge of the underlying architecture such as the size of cache lines and victim caches may also help in writing better and ultimately faster code