This lab investigates the use of multiple threads in programs. If a two-thread program is run on a system with two processors, both threads could actually be running at the same time. Of course, if a two-thread program is run on a single CPU system the threads will only appear to be concurrent.

1. Get copies of the program `argus.c` which is a 2-thread simulation of a dog (argus) with main thread (“Will you please throw my ball?”) and subconscious thread (“sleeping..eating..scratching..”). Test this program. Argus will keep asking you to throw his ball until you tell him “all_done.” Note that `argus` cannot be stopped with a CTRL-C (you will have to enter “quit”) since the signal SIGINT is trapped. When Argus’ subconscious thread deactivates it notifies the main thread with the signal SIGUSR1. On a multiple CPU system it may well be the case that the two threads are at times running on different CPU’s.

2. Some systems (e.g. Sun’s Solaris) have limitations in delivering signals to multi-threaded programs. See `argus_mutex.c` for another way to write the program which uses a mutex and a condition variable rather than the signal SIGUSR1.

**Assignment** Write a two-thread program which first generates a 250x250 matrix \( \{m_{ij}\} \) of random double precision real numbers chosen from \([-1.0, 1.0]\). So one can use

\[
x = 2.0 \times (((\text{double})\text{rand()}/(\text{double})\text{RAND\_MAX}) - 0.5);
\]

Then the program computes the square of this matrix, i.e. the \( ij \)-th entry \( \{d_{ij}\} \) of the squared matrix is

\[
d_{ij} = \sum_{k=0}^{249} m_{ik}m_{kj}
\]

The program should divide up the task equally between the two threads. Note that the two threads could have the same code, but just accept different parameters. When you write this program you should first try a small matrix (say 4x4) to test that the code is working properly.