This lab concerns user processes and what the important attributes and parameters attached to a user process are (from the point of view of the operating system).

1. Get copies of the sample program `process.c` and the makefile (if necessary, copy `makefile.unx` to `Makefile`) in one of your subdirectories. Compile and link the program by typing:
   ```
   make process < cr >
   ```

2. Try running the program. It will prompt you for the info-flags you wish to set and you can enter the octal code with leading zero (or you can put the info-flags as the first argument on the command line). You may wish to copy the output to a file via:
   ```
   process [info-flags] 2>process.out < cr >
   ```
   Note that the output of the program is written to stderr (descriptor 2) not stdout (descriptor 1). This is so that you can run it interactively (with stdin and stdout) while it writes to stderr. Note that some information might not be obtained and marked with “–na–.”

3. Open a second xterm (or console) and run the program again but let the program block when is says “Pausing . . .”. On the other xterm, use the `ps` command with a variety of options (e.g. `aux` (BSD), `avx` (BSD), `-Aef` (SYSV), `-Aely` (SYSV), . . ., `as` (DEC)) to try to find out the process information manually. One useful option with the (SYSV) command is `ps -[f | ly | j]u [user | uid]`. Write all you can find down, finish the program and compare information. You can also get the memory map with the command `pmap -x [process_id]`. The `ps` command can find some things you can’t as a user because it is allowed to read kernel memory (`/dev/kmem`). You should especially make sure you can find the following (note that there are differences between systems in both the abbreviations and the units of measurement – when in doubt, rtfm: `man ps`):

   - **UID, PID, PPID**
   - **TTY or TT** (controlling terminal, if any)
   - **RSS** (resident set size)
   - **VSZ or SZ** (virtual size)
   - **WCHAN** (address of event for which process is waiting)
   - **PRI** (priority in system-dependent units)
   - **CMD** (command line invocation via `execve()`)
   - **STAT or S** (process status: R-runnable, T-stopped, S-sleeping, Z-zombie, . . .)
   - **STIME** (starting time)
   - **TIME** (profiling time - user plus system)