Processes and Job Control
Unix is a multi-tasking operating system
- some of these tasks are being done by other users logged in
- some are being done by you in the background
  - e.g. watching for incoming mail
When you run a task (a Unix command, like `ls` or `vi`) it executes in the foreground of your shell
  - it has the “control” of your screen and keyboard
Foreground and Background (2)

❖ If you still want to use the current shell
   ❖ obelix[1] > a_heavy_task &
   ❖ [1] 13607
   ❖ obelix[2] >

❖ When you put a task in background
   – task keeps running, but you continue to work at the shell in the foreground
   – if any output is done, it appears on your screen immediately (can be confusing)
   – if input is required, process prints a message and stops
   – when it is done, a message will be printed
Foreground and Background (3)

- Explicit background processes are needed less often with windowing systems
  - Just go to another window and run the command

- But explicit background processes are used often in unix
  - A command needs a long time, you do not want to close that window by accident
  - Run a job at the background and logout
  - `netscape&` will open a new window, but leave the current shell window still available to use
A Simple Script

- We use the following shell script to illustrate job control
- Edit a file make_noise
  ```
  obelix[1] > cat > make_noise
  #!/ bin/ sh
  while [ 1 ]
  do
    date
    sleep 1
  done
  obelix[2] > chmod u+x make_noise
  ```
- `make_noise` then is a shell script repeats to print the time for every second, until you terminate it using Ctrl- c.
Job Control – Suspending Jobs

- csh, tcsh, and bash allow you to manage the running of different processes

- Suspending jobs
  - the Ctrl-z special character stops the job

```
obelix[1] > make_noise
Fri May 16 14:14:43 EDT 2003
```

```
......
^Z
Suspended
```
```
obelix[2] > vi readme
^Z
```
The "jobs" command shows which of your jobs are running and/or stopped.

```
obelix[3] > jobs
[1] + Suspended     make_noise
[2] + Suspended     vi readme
```

Here there are two suspended processes, the make_noise and a vi process.
Job Control – Resuming Jobs

- Putting jobs back into the foreground:
  - Use the "fg" command to move a job into the foreground.
    \[ \text{obelix}[4] > \ fg \ %2 \]
  - Puts job number 2 into the foreground.
  - Works with either a background or stopped job.

- Putting jobs into the background:
  \[ \text{obelix}[5] > \ bg \ %1 \]
Job Control – Killing Jobs

❖ Jobs can also be killed
   – Use the Unix "kill" command
     obelix[6] > kill %1
     or if it won't die ...
     obelix[7] > kill –9 %1

❖ Jobs can be stopped and continued
  obelix[8] > a_heavy_task &
  obelix[9] > stop %1
  obelix[10] > bg %1
Using ps (1)

- Jobs are really just a special case of Unix processes
- `ps` can list the current processes
  
  ```
  PID   TT    S TIME COMMAND
  2312 pts/0  T 0:00  vi
  2296 pts/0  R 0:00  tcs
  2313 pts/0  R 0:00  ps
  ```

- `ps` can take many options, depending on which version of `ps` you are using (`/usr/bin/ps` vs. `/usr/ucb/ps`)
Using `ps` (2)

- The `ps` command takes a number of options
  - `-l` gives you a long listing of what is going on
  - `-u loginid` tells you about loginid's processes
  - `use man ps` to see more options
- `kill pid` kills the process pid
  - `TERM` signal will be sent to the process pid
  - `kill -9` or `kill -KILL` will send the `KILL` signal
  - Use `man kill` to find out more signals
Another useful command: ulimit

- The **ulimit** utility sets or reports the file-size writing limit imposed on files written by the shell and its child processes (files of any size may be read). Only a process with appropriate privileges can increase the limit.
  - `-a` prints all limits
  - `-f` maximum file size (in 512-byte blocks)
  - `-v` maximum size of virtual memory (in kbytes)

- Let us illustrate the interest of ulimit
  
  [moreno@iguanodon shell]$ ulimit -u 100
  [moreno@iguanodon shell]$ more foo
  echo FOO
  ./bar
  [moreno@iguanodon shell]$ more bar
  echo BAR
  ./foo
  [moreno@iguanodon shell]$ ./foo