# Hands-on 2: Time Sharing System

Michael Plasmeier

1. ps | tail –n +2 | sort –k4d
2. egrep -v "a|e|i|o|u" /usr/share/dict/words | wc –w
3. echo -e "1 0 1 0 1 \n0 1 0 1 0 \n1 0 1 0 1 \n0 1 0 1 0 \n1 0 1 0 1 \n0 1 0 1 0"
4. ls -lS /etc | grep ".conf" | tail -n 5 | sort -k5n,5
5. The second command fails if you do not have write permission in the directory. In addition, information could be stale by the time the two commands are run (race condition). The temp file might not get deleted, leaving problems in the future. If someone else has a temp file they forgot to delete, then you can’t delete it and you need to alter your command.
6. Yes, this is the expected behavior. In the first example with ; the computer is running the file once completely y n and then running the other file y n. In the other & case, the computer is running each file simultaneously. When the processor sees the sleep command, it runs the other file, leading to y y n n.
7. The Unix system maintains an internal state that keeps track of a position pointer. This makes sequential writing to a file a simple exercise. I’m not sure why this choice was made. The offset could change based on other people writing to a file, so the Unix designers wanted people to read and verify to move to a certain position in the file.

If an application wanted to move to a specific offset in a file, the application could call lseek with SEEK\_SET