Lecture 5 — Outline

Topics: Devices and Files

Approach: Ideas, Examples, Applications
Featuring: write, stty

Main Ideas:

Device I/O works just like disk file I/O
Devices have names, properties, contents, use open, read, write
Devices work differently from disk files
Connections to disk files have specific attributes
Connections to terminal files have specific attributes
stty is a command that allows one to examine/change attribs

Outline

The story so far
  We know how to read, write, create, rename, remove files

Questions for tonight
  What about devices: terminals, printer, scanners, sound
  Is programming for devices like programming for files?
  How can a programmer gain access to and control devices?

Device I/O is just like Disk File I/O
  open, read, write, close, lseek can all be used
  Every device has a name in the file system
  examples: who > /dev/ttyp2 ; od -c /dev/ttyp2
  Every device has attributes (ls -l /dev/xxx)
  application: the write command, our versions

Device I/O is nothing like Disk File I/O
  Disk files use buffering; terminals cannot use buffering
  Terminals have baud rate, parity, echo, buffering...

Attributes of file descriptors: using fcntl and open
  buffering can be turned on or off (O_SYNC)
  auto-append can be turned on or off (O_APPEND)
  using fcntl()

The kernel processes data going to and from terminals
  The kernel software is called the *terminal driver*
  The terminal driver can do lots of things and has many settings
  The stty command is the user interface
  The tcgetattr() and tcsetattr() are the programmer interface

Bits on Parade
  How to (1) test a bit, (2) set a bit, (3) clear a bit
  echostate, setecho

Writing stty