CSE 506: Operating Systems

Terminals

Terminals

- Kernel structure
  - ***Terminals***
    - Connect to "real" devices
  - ***Pseudo-terminal (pty)***
    - Connect to processes pretending to be real devices
    - E.g., xterm or sshd
- Terminals connect to drivers on one end
  - Driver provides input and output functionality
    - Output: output a character (or string)
    - Input: get a character
  - Modern OSes have console driver with multiple terminals
    - Switch between terminals with Alt+F? keys (***virtual*** terminals)

Console Driver

- `output` function outputs to screen
  - Similar to printk()
  - How to distinguish printk() from terminal?
    - printk() is usually bold
  - Needs to support printing special characters
    - E.g., "go back character" goes left 1 position
- `input` functionality gets input from keyboard
  - Translates scan codes to characters
    - Actually integers, to allow for non-ASCII chars
      - Meta (Alt), arrows, etc…
  - Calls input function on terminal

Terminal Input Processing

- When receiving a new character…
  - Handle control characters
    - Ctrl+H: backspace
    - Ctrl+U: kill (clear buffer)
    - Ctrl+S: stop (buffer output), Ctrl+Q: start
  - Perform local echo
  - Keep buffering up input
    - If Enter (Ctrl+M) received, try to pass input (called ***cooked*** mode)
- What happens when buffer is full?
  - Beep and drop characters
  - Bad idea to drop from head of buffer

Application Interface

- Apps interact with terminals via file descriptors
  - When app starts, kernel auto-opens 3 descriptors
    - stdin (0), stdout (1), stderr (2)
  - Descriptors are connected to terminal
- On `write` syscall from app
  - Terminal output function is called
    - Passed output to driver (unless “stop”ed)
- On `read` syscall from app
  - Terminal input function is called
    - If buffer contains newline, process `read` call
    - If buffer has no newline, block (terminal will unblock on Enter)

Terminal Inheritance

- How do `fork()`ed procs interact with terminal?
  - It's complicated – too much to cover here
- The important bits for CSE506…
  - Child processes connect to same terminal
    - Members of same ***session***, split into ***groups***
    - Syscalls for creating new sessions, new groups/leaders
      - Not required for SBUnix
  - Output from all children goes to terminal
  - Input from terminal goes *only* to "foreground" process
  - If non-fg processes try to `read`, receives SIGTTIN

Lots of Cool Features

- Ctrl+C: send interrupt signal to foreground group
- Ctrl+Z: send suspend signal to foreground group

- In reality, terminals are highly configurable
  - See **`stty -a`** on Linux system

```
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```