Zajímavosti ze systémového programování

Pavel Šimerda

pavlix@pavlix.net

C API

File I/O

- Let's start with shell
- Use `cat` to read a file
- Use `strace` to trace the syscalls
- Use `ltrace` to trace the library calls
- Call ANSI `fread` from a binary
- Call POSIX `read` from a binary

fopen + fgets

- ANSI file I/O
- Returns pointer to buffer on success
  - No use for the pointer, though.
- Returns NULL on error
- Returns NULL on end of file
- Stops reading at end of line
- Supplies the cstring NUL character
- Doesn't work with binary data

open + read

- POSIX file I/O
- Almost 1:1 with Linux syscalls
- Returns positive number on success
  - Number of bytes actually written
- Returns zero on end of file
- Returns minus one on error
  - Check errno for details
- Doesn't care about new lines
- Works with text and binary

Memory Allocation and Reallocation

- Static global allocation
- Stack local allocation
- Dynamic allocation and reallocation
- Reallocation strategy

Process Management

- Kernel tasks
  - Processes
  - Threads
  - Namespaces

fork

- Library function
  - Uses `clone` syscall
- Returns 0 in child process
- Returns child pid in parent process
- Returns -1 on error
  - Check errno

exec

- Set of library functions
  - `execlp()` for example
  - Using `execve()` and `PATH`
- Doesn't return on success
- Returns -1 on failure
  - Check errno
- File descriptor handling
  - `O_CLOEXEC`
  - What if std streams were closed?

Process Creation Strategy

- Master process and forked processes
  - Easy to create shared resources
- Thread creation
  - Many more resources shared by default
- Separately forked processes
  - Identifying shared resources via name or path
  - Passing shared resources via IPC
- Service management and environment
  - initscripts
  - systemd and other service managers

Pipes and Sockets

- Communication
- Forked pipe and socketpair
- Named pipes and unix sockets
  - Run `mkfifo` binary
  - `mkfifo()` function
  - `mknod` syscall

Shared Memory and Mutual Exclusion

- Consumer-producer example
- Just mmap()
- Path based shared memory
- Path based semaphores

Asynchronous Signal Handling

- Classic async handlers
- Returning control
- Interrupted syscalls
- Event based handling
  - `signalfd`

Bonus: Debugging and Tracing

- `strace` the `strace`!
- Example debugging and tracing sessions
- Tracing those to see how they use the operating system

Bonus: Network Sockets

- Using TCP/IP sockets
- Dual stack socket abstraction
  - `multisock`
- Name resolution
  - `getaddrinfo` call
  - `netresolve` project

Bonus: I/O Multiplexing

- `select`, `poll`, `epoll`
- User space implementations
  - `userspacefd` project