Sistemi Operativi, Secondo Modulo, Canale M–Z
Riassunto della lezione del 15/05/2017

Igor Melatti

Il Linguaggio C

- Slides da `https://www.cs.clemson.edu/course/cpsc111/slides/`; vederle tutte, ma qui ci concentriamo su quelle più importanti
- Capitolo 8 (fine), 9 e 16

Le chiamate di sistema (*system call* o *syscall*)

- Considerazioni sul `man`
  - prototipo, sezione 2
  - le system call sono quelle in `apropos -s 2 .`
- Le "vere" chiamate alle syscall andrebbero fatte come descritto in `man 2 syscall` e `/usr/include/x86_64-linux-gnu/sys/syscall.h`
  - andrebbero fatte in assembler perché, per motivi di sicurezza, le system call usano uno stack a parte
  - per semplificare queste chiamate, si usano dei wrapper, che sono quelli della sezione 2
  - alcuni sono wrapper di wrapper, nel senso che chiamano un wrapper anziché la syscall "vera"
  - per lo più, hanno lo stesso nome, o quasi, della corrispondente syscall; in alcuni casi, ci sono più versioni della stessa syscall, essenzialmente la differenza è che viene settato un default per alcuni argomenti
  - le syscall vere hanno sempre il numero massimo di argomenti
  - contengono delle syscall anche alcune funzioni della libreria standard (che sono invece nella sezione 3 del `man`), come quelle che hanno a che vedere con i file
  - contengono delle syscall anche `malloc, calloc, realloc, free` ed `alloca`, che chiamano la `sbk`; inoltre, tail funzioni non sono solo wrapper, ma hanno anche delle strutture dati per permettere la `free` di singole variabili

  - diversamente dalle funzioni della standard library (es. printf, fopen, ...) le syscall *non* sono portabili: non funzionerebbero (o meglio, non è garantito che siano definite), ad esempio, sotto Windows

- In caso di errore, i wrapper settano anche `errno`, e una spiegazione dell'errore può essere ottenuta chiamando le funzioni (non di sistema) `strerror` e `perror` (vedere `man errno` e l'esempio `strerror.c` allegato)
  - attenzione: non fare la free del risultato di `strerror`! Si tratta di una string statica, e non allocata con `malloc` o simili
- Comando `strace`, permette di vedere tutte le syscall effettuate, sia di processi ongoing che di comandi da passare a `strace` stesso
  - vedere `man strace`

Le syscall per la gestione dei file

- Standard e portabili: `fopen`, `fclose`, `fseek`, `ftell`, `rewind`, `fgets`, `fread`, `fwrite`, `fscanf`, `fprintf` (capitoli 15 e 16)
- Syscall, disponibili solo sotto Linux: `open`, `close`, `lseek`, `read`, `write`
  - niente output o input formattato (a meno di non bypassare il tutto con `fdopen`)
  - niente `rewind` ed `ftell`, che vanno realizzati con `lseek`
- Altre syscall sui file, senza corrispettivo sulla libreria standard: `chown`, `chmod`, `stat`, `lseek`, `select`, `ioctl`, `unlink`, `fcntl`, `symlink`, `link`, `chdir`, `rename`, `mkdir`, `rmdir`, `chroot`
- Le syscall della libreria standard sono applicabili solo a file regolari; quelle di Linux tengono conto di tutti i tipi di file disponibili (in Linux, tutto ciò che non è un processo è un file...)
  - directory
  - devices
  - pipe
  - fifo
  - socket
- A parte apertura e chiusura, le operazioni non includono solo lettura, scrittura e posizionamento, ma anche operazioni speciali
- Vedere gli esempi allegati (sono corredati di commenti ed esercizi), in quest'ordine: `open.c`, `open_read_write.c`
- Per ogni syscall menzionata, leggere il `man`