



Sistemi
Operativi

Bruschi
Monga Re

Lezione V: Shell 1

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Sistema Operativo

Un s.o. è un programma che rende conveniente l'uso dello hardware

- fornendo astrazioni che semplificano l'uso delle periferiche e della memoria
- gestendo opportunamente le risorse fra tutte le attività in corso

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



- Useremo un Live CD: Debian GNU/Linux (<https://debian-live.alioth.debian.org/>)
- Personalizzato per il corso, contiene:
 - busybox
 - nasm
 - gcc
 - binutils
 - make
 - git
 - gdb
 - Utilità di rete: openssh-client, dropbear, rsync
 - Più avanti utilizzeremo una parte *persistente* per gli esercizi JOS.
- Tutti programmi *console-based* per risparmiare spazio e permetterne l'uso anche in condizioni di risorse limitate

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programma-
tore e il
s.o.

Astrazioni

Editor

Chiamate
implicite

Esercizi



- Il Live CD è utilizzabile nativamente o con una macchina virtuale qualsiasi (VirtualBox, VMware, ecc.)
- Gli esercizi però sono provati con Qemu (<http://www.qemu-project.org/>)
 - i440FX host PCI bridge and PIIX3 PCI to ISA bridge
 - Several video card (VGA)
 - PS/2 mouse and keyboard
 - 2 PCI IDE interfaces with hard disk and CD-ROM support
 - Floppy disk
 - Several network adapters (Intel e1000)
 - Serial ports
 - PCI UHCI USB controller and a virtual USB hub.

Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programma-
tore e il
s.o.

Astrazioni

Editor

Chiamate
implicite

Esercizi



Ada, che ha a disposizione una macchina i386, vuole scrivere un programma che calcoli la somma di 42 e 24 e conservi il risultato in una specifica cella di memoria.

Sostanzialmente:

```
segment .text ; segmento "testo" (istruzioni)
global main ; nome convenzionale
```

```
main: mov eax, 24
      add eax, 42
      mov [x], eax
```

```
segment .data ; segmento dati
x: dw 0 ; pseudo istruzione: double word (2 byte)
```

Nasm micro bigino:

<https://www.cs.uaf.edu/2006/fall/cs301/support/x86/>

Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Per risolvere il suo problema Ada *deve* fare uso delle **astrazioni** fornite dal s.o. perché l'accesso diretto allo *hardware* è interdetto. Tipicamente:

- *System call*
- Memoria virtuale
- Programma in esecuzione: **Processo**
- Persistenza: *File*
- *Shell* (interprete comandi)

L'insieme di queste costituisce una **macchina virtuale** piuttosto differente dal dispositivo elettronico i386.

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Di cosa ha bisogno?

Ada *vuole* **scrivere** il suo programma in *assembly*.

- scrive attraverso un **programma** (*editor*)
- ciò che scrive deve **persistere anche al termine dell'esecuzione dell'editor** (*file*)
- un altro programma (*assemblatore*) traduce il programma in linguaggio macchina (e di nuovo deve persistere)
- **esegue** il programma in linguaggio macchina

Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



File

Un *file* è una sequenza di byte conservati in maniera persistente rispetto all'esecuzione dei programmi. Alla sequenza è associato un nome e altri attributi.

Nei sistemi *unix-like* i *file* sono organizzati gerarchicamente in *directory* (l'equivalente dei *folder* di MS Windows), che non sono altro che *file* che contengono un elenco di 'nomi' (in realtà come vedremo *i-node*).

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Digressione: editor (di testo)

Editor

Un *editor* è un programma che permette di modificare arbitrariamente un *file*. Un *editor* di testo generalmente manipola *file* composti da caratteri stampabili (ASCII \rightsquigarrow 1 byte, UTF-8 \rightsquigarrow da 1 a 4 byte).

- Emacs (zile), **vi**, ne,...
- Notepad, Textpad,...

Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma
e il
s.o.

Astrazioni

Editor

Chiamate
implicite

Esercizi



Bill Joy (co-fondatore della SUN), 1976, per BSD UNIX

- *Modal editor*
 - modo input
 - modo comandi
- I comandi di movimento e modifica sono sostanzialmente *ortogonali*
- small and fast
- fa parte dello standard POSIX

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programma-
tore e il
s.o.

Astrazioni

Editor

Chiamate
implicite

Esercizi



Salvare un file e uscire wq

- Modifica:
 - i, a insert before/after
 - o, O add a line
 - d, c, r delete, change, replace
 - y, p "to yank" and paste
 - u undo . redo
 - s/reg/rep/[g] search and replace
- Movimento:
 - h, j, k, l (o frecce)
 - 0, beginning of line, \$, end of line
 - w, beginning of word, e, end of word
 - (num)G, goto line num, /, search
 - (,), sentence

Tutorial <http://openvim.com>.

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Shell

La *shell* è l'*interprete dei comandi* che l'utente dà al sistema operativo. Ne esistono grafiche e testuali.

In ambito GNU/Linux la piú diffusa è una shell testuale `bash`, che fornisce i costrutti base di un linguaggio di programmazione (variabili, strutture di controllo) e primitive per la gestione dei processi e dei *file*.

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



```
# tramite la shell ordina l'esecuzione di vi  
# (argv[0] "vi") con parametro argv[1] "somma.asm"  
vi somma.asm
```

Perché un programma possa essere eseguito deve essere in un formato (convenzioni) comprensibile al s.o. (p.es. ELF per Linux)

```
# tramite la shell ordina l'esecuzione di nasm  
# parametro argv[1] "-f" argv[2] "elf" argv[3] "somma.asm"  
nasm -f elf somma.asm -o somma.o  
# collegamento del file oggetto in un eseguibile  
# argv[1] ... argv[3]  
gcc -o somma somma.o
```

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programmatore e il
s.o.

Astrazioni

Editor

asm
Chiamate
implicite

Esercizi

Ada ha risolto?



Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi

Per il momento Ada può vedere il risultato solo tramite l'esecuzione del suo programma tramite un *debugger* (il quale chiede al s.o. di eseguire un altro programma e tenerlo 'sotto controllo').

GDB microbigino

<http://www.cs.mcgill.ca/~consult/info/gdb.html>

```
# tramite la shell ordina l'esecuzione di gdb  
# con parametro argv[1] "./somma"  
gdb ./somma
```

Per stampare il risultato deve **necessariamente** fare uso di **system call**



Che succede se Ada scrive nella memoria su cui è mappata la scheda video?

```
mov ebx, 0xb8000
mov ds, ebx
mov byte [10], 'm' ; indirizzamento relativo a ds
```

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programma-
tore e il
s.o.

Astrazioni

Editor

Chiamate
implicite

Esercizi



Una chiamata di sistema (**syscall**) è la richiesta di un servizio al sistema operativo, che la porterà a termine in conformità alle sue *politiche*.

Per il programmatore è analoga a una chiamata di procedura. Generalmente viene realizzata con un' *interruzione software* per garantire la protezione del s.o..

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Un'interruzione (*interrupt request (IRQ)*) è un segnale (tipicamente generato da una periferica, ma non solo) che viene notificato alla CPU. La CPU, secondo le politiche programmate nel PIC, risponderà all'interruzione eseguendo il codice del *gestore dell'interruzione (interrupt handler)*. Dal punto di vista del programmatore la generazione di un'IRQ è analoga ad una chiamata di procedura, ma:

- Il codice è completamente disaccoppiato, potenzialmente in uno spazio di indirizzamento diverso (permette le protezioni)
- Non occorre conoscere l'indirizzo della procedura
- La tempistica dell'esecuzione è affidata alla CPU

In Linux a 32bit <https://www.cs.utexas.edu/~bismith/test/syscalls/syscalls32.html>

Sistemi Operativi

Bruschi
Monga Re

Le astrazioni del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programmatore e il s.o.

Astrazioni
Editor

Chiamate implicite

Esercizi

Asm syscall



```
segment .text
```

```
global main
```

```
main:
```

```
    mov ecx, msg           ; stringa
    mov edx, msg_size     ; dimensione stringa
    mov ebx, 1           ; file descriptor (stdout)
    mov eax, 4           ; syscall 4 (write)
    int 0x80

    mov eax, 1           ; syscall 1 (exit)
    int 0x80
```

```
segment .rodata
```

```
msg: db 'Ciao solabbisti!',10,0
```

```
msg_size equ $ - msg
```

Ada può far meglio. . .



Stampare il risultato direttamente con la system call è piuttosto oneroso: p.es. occorre occuparsi di convertire il numero risultante nei caratteri corrispondenti alle sue cifre decimali. La **libreria** del C contiene una funzione `printf` che semplifica molto il lavoro di Ada. . .

`extern printf`

```
push .... ; parametri sullo stack (push in ordine inverso)
call printf ; valore di ritorno in eax
```

Serve anche una chiamata alla libreria `exit` (con parametro 0) perché la `printf` è “bufferizzata” e la stampa vera e propria avviene solo all’uscita del programma (o in altri momenti in cui vengono “svuotati” i *buffer*).

Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programmatore
e il s.o.

Editor

Chiamate
implicite

Esercizi



Sistemi
Operativi

Bruschi
Monga Re

- 1 Perfezionare il programma di Ada in modo che stampi il risultato
- 2 Scrivere in assembly un programma che saluta l'utente dopo averne chiesto il nome
- 3 Scrivere in assembly un programma che stampa la somma di due numeri interi
- 4 Scrivere in assembly un programma che stampa il fattoriale di un numero passato come parametro

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi



Sistemi
Operativi

Bruschi
Monga Re

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio
Qemu

Il programma-
tore e il
s.o.

Astrazioni
Editor

Chiamate
implicite

Esercizi

- Edsger W. Dijkstra, “My recollections of operating system design” <http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1303.PDF>



**Sistemi
Operativi**

**Bruschi
Monga Re**

Le astrazioni
del s.o.

Il ruolo del s.o.

Setup del laboratorio

Qemu

Il programma-
tore e il
s.o.

Astrazioni

Editor

**Chiamate
implicite**

Esercizi