



Sistemi Operativi¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2015/16

¹ © 2008–16 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>. Immagini tratte da [2] e da Wikipedia.



Lezione VIII: Shell 2



Astrazioni fornite dal s.o.

Per risolvere il suo problema Ada *deve* fare uso delle astrazioni fornite dal s.o.. Tipicamente:

- *System call*
- Memoria virtuale
- Processo
- *File*
- *Shell*

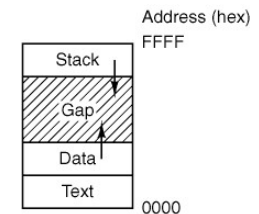
L'insieme di queste costituisce una macchina virtuale piuttosto differente dal dispositivo elettronico i386.



Memoria virtuale

Il programmatore è libero di considerare un unico spazio di memoria, interamente dedicato al suo programma. Questo spazio può anche essere superiore alla memoria fisicamente disponibile.

Generalmente la memoria virtuale è divisa in *segmenti*: testo (codice), dati inizializzati, stack e heap.



Processo



Sistemi
Operativi

Bruschi
Monga Re

Astrazioni

Programma

Un programma è la codifica di un algoritmo in una forma eseguibile da una macchina specifica.

Processo

Un processo è un programma in esecuzione.

Thread

Un thread (*filo conduttore*) è una sequenza di istruzioni in esecuzione: più thread possono condividere lo spazio di memoria in cui le istruzioni lavorano. Il termine assume anche un'accezione tecnica nei sistemi operativi che distinguono le due astrazioni.

Ogni processo dà vita ad **almeno** un thread. Ogni CPU in un dato istante può eseguire **al più** un thread.

172

POSIX Syscall (process mgt)



Sistemi
Operativi

Bruschi
Monga Re

Astrazioni

<code>pid = fork()</code>	Create a child process identical to the parent
<code>pid = waitpid(pid, &statloc, opts)</code>	Wait for a child to terminate
<code>s = wait(&status)</code>	Old version of waitpid
<code>s = execve(name, argv, envp)</code>	Replace a process core image
<code>exit(status)</code>	Terminate process execution and return status
<code>size = brk(addr)</code>	Set the size of the data segment
<code>pid = getpid()</code>	Return the caller's process id
<code>pid = getpgrp()</code>	Return the id of the caller's process group
<code>pid = setsid()</code>	Create a new session and return its process group id
<code>l = ptrace(req, pid, addr, data)</code>	Used for debugging

173



Sistemi
Operativi

Bruschi
Monga Re

Astrazioni

410