# Laboratorio di Sistemi Operativi

**II Semestre - Marzo/Giugno 2007**

Prof. Ferdinando Cicalese
Dipartimento di Informatica ed Applicazioni
Università di Salerno
cicalese@dia.unisa.it

## Orario del corso
### Classe 1

- Lunedì 16:00 – 18:00 P-3
- Mercoledì 16:00 – 18:00 Lab. Reti
- Giovedì 15:00 – 17:00 Lab. Reti (classe A)
- Venerdì 13:00 – 15:00 Lab. Reti (classe B)

- Classe A={matricole congrue 0 modulo 6}
- Classe B={matricole congrue 3 modulo 6}



## Orario di ricevimento
### (da confermare)

- Lunedì 15:00 – 16:00
- Mercoledì 15:00 – 16:00



## Testi di riferimento

- [1] W. Richard Stevens, Stephen A. Rago
  "Advanced Programming in the Unix Environment"
  Addison Wesley, 2nd edition

- [2] C. Newham, B. Rosenblatt
  "Learning the bash"
  O'Reilly, 3rd edition

## Ulteriori riferimenti per Linux

- Linux Documentation Project:
  - Linux User's Guide
- M. Sobell
  "A practical Guide to Linux", Addison Wesley
- *Guida dell'utente di Linux*,L. Greenfield, ftp://ftp.pluto.linux.it/pub/pluto/ildp/guide/Guida-Utente (in italiano)
- *Bash Reference Manual*, Free Software Foundation http://www.gnu.org/software/bash/manual/bash.html



## Sistema Operativo

- Unix
- Linux
- Mac OS X - Darwin
- TARGET: avere un'alternativa al "classico" PC



## Compilazione

- Il libro di Stevens utilizza per gli esempi una speciale libreria di funzioni: apue
- I file e le informazioni necessarie per utilizzarli saranno rese disponibili in linea nella pagina dedicata al corso all'indirizzo
  - http://www.dia.unisa.it/professori/cicalese/LSO



## Argomenti del Corso

- Standard UNIX e Implementazioni
- I/O "*non bufferizzato*"
- File e directory
- Cenni di librerie di I/O standard
- Processi
- Cenni sui segnali
- Interprocess Communication
- Introduzione alla programmazione shell: bash

## Pagina web

- Far riferimento alla pagina web
  http://www.dia.unisa.it/professori/cicalese/LSO
  - Programma (...aggiornato di volta in volta)
  - Slide delle lezioni
  - Informazioni sul corso



## Introduzione

## Introduzione: Architettura UNIX

- Ogni s.o. fornisce servizi ai programmi che girano
  - esecuzione di nuovi programmi
  - aprire/chiudere file
  - allocare memoria
- Noi studieremo i servizi offerti da UNIX
- Descrivere un sistema operativo è impossibile senza introdurre i suoi "termini" salienti



## Introduzione: Accesso al Sistema

- Logging In:
  - L'accesso a UNIX prevede l'inserimento di login e password
  - Login: cicalese
    password: *******
  - Il sistema controlla che la coppia login-pwd sia registrata nel file delle password (tipicamente /etc/passwd)

    cicalese:x:102:107:Ferdinando Cicalese:/home/cicalese:/bin/bash

## Identificatori utenti

- Un utente viene identificato dal sistema attraverso:
  - User Id : valore numerico che ci identifica in /etc/passwd
  - Group Id : nel file /etc/group c'è la mappa tra i nomi dei gruppi ed i rispettivi identificatori numerici
- superuser ha uid = 0

Esempi:

ls –l usa /etc/passwd per printare la login name del proprietario

```
printf("uid =%d, gid =%d\n",getuid(),getgid());
```



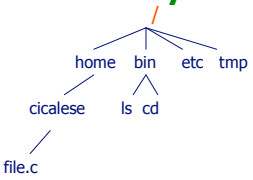
## Introduzione: Programmi Shell

- shell: interprete di comandi
  - sh (include reminiscenze di ALGOL)
  - csh (usa una sintassi simile al C, prende comandi interattivi o da programmi, job control, history,…)
  - tcsh (versione avanzata di csh, completamento automatico dei comandi, viaggiare su e giù per la lista dei comandi dati, spelling correction dei comandi )
  - bash (estende sh ereditando anche da csh, shell programming più facile che in tcsh )



## Introduzione: File e Directory

- Filesytem gerarchico
  - files & directory
  - root /




- Pathname: sequenza di *zero* o più nomi di file separati da /

Es: /home/cicalese/file.c
/ *(directory root)*

- Nomi di file speciali:
  - dot .
  - dot dot ..



## Introduzione: I/O

- Descrittori di File: intero positivo (piccolo) usato associato ad un file aperto. Viene poi usato in tutte le successive operazioni di lettura e scrittura.
- Standard Input, Output, Error: i descrittori 0,1,2, aperti per default da qualsiasi programma in esecuzione (rappresentano il terminale)
- Unbuffered I/O: funzioni che lavorano con descrittori di file. Ogni operazione chiama una *system call*.
- Standard I/O: per es. la maggior parte delle funzioni di I/O a voi gia' note, come *printf*

## Introduzione: Processi

- Programma: file eseguibile
  - letto dalla memoria
  - eseguito dal kernel
- Processo: istanza in esecuzione di un programma (detto anche task)
- Ogni processo ha un identificatore numerico pid>0 associato mediante il quale ci si riferisce



## Introduzione: Gestire gli Errori

- quando avviene un errore in una funzione UNIX, viene restituito un intero negativo (-1, in genere)
- la vbl intera errno è settata ad un valore che fornisce ulteriori info
- vedi la sezione 2 del manuale di UNIX: man 2 intro
- <errno.h> contiene i valori costanti assumibili da errno
  - In linux: </usr/include/errno.h>



## Introduzione: segnali

- Utili per segnalare ad un processo che qualche condizione di interesse si é verificata.
  Es:
  - se il processo effettua divisione per zero, il segnale SIGFPE gli viene inviato dal sistema.
  - Il processo può: ignorare il segnale, lasciare che una qualche azione di default sia intrapresa, fornire una funzione propria per gestire il segnale.



## System Call & librerie

- Una system call è un entry point del kernel per fornire servizi ai processi che li richiedono
- man 2 fornisce la documentazione sulle sistem call (definite in C)
- System call → funzione omonima nella libreria standard (wrapper)
  - Es. ssize_t read (int *filedes*, void **buff*, size_t *nbytes*);
- l'utente chiama il wrapper (attraverso la sequenza standard di chiamate a funzioni di C), questo invoca il servizio del kernel
- Semplifichiamo: System Call = Funzioni C

## System Call & Librerie

- man 3 contiene le funzioni general purpose per il programmatore (libc)
  - printf, malloc, etc.
  - non sono system call

Esempio:
il codice vuole allocare memoria → invoca malloc la quale per allocare realmente utilizza la system call sbrk



## System Call & Librerie

- system call: interfaccia minima, mentre le funzioni di libreria forniscono più elaborate funzionalità
- libc: interfaccia normale
- system call sono definitive dal s.o., le funzioni di libreria no!



## Standard Unix & Implementazioni

## Standard Unix: ANSI

- American National Standards Institute: venditori + utenti
- Membro dell' International Organization for Standardization
- 1989-1999: ANSI C/ISO C standardizzazione del linguaggio C
  - portabilità di programmi C ad una grande varietà di S.O. e non solo per UNIX
  - Definisce non solo la semantica e la sintassi ma anche una libreria standard divisa in 24 aree (individuate dagli header, vedi fig. 2.1 di [1])

## Standard Unix: IEEE POSIX.1

- Institute of Electrical and Electronic Engineers propose una famiglia di standard
  - Portable Operating System Interface
  - Lo standard 1003.1 relativo a interfacce di s.o.: definizione di servizi che un s.o. deve fornire per essere POSIX COMPLIANT
    - Definisce una interfaccia non una implementazione
    - Non è fatta distinzione tra system call e funzioni di libreria
    - Non prevede la figura di "superuser", ma certe operazioni richiedono appropriati privilegi



## Standard Unix: SUS (Single Unix Specification)

- … dell' X/Open: gruppo di venditori di Computer
- Open group pubblicava la X/Open Portability Guide
  - selezionare e migliorare gli standard sul mercato
  - XPG3 - presenta un nuovo standard basato su POSIX
- XSI - X/Open System Interface
  - Definisce le parti opzionali di POSIX.1 necessarie per ottenere XSI-conformity
  - Un sistema può dirsi UNIX solo se XSI-conforming



## Implementazioni: SVR4

- System V Release 4 è stato prodotto dalla AT&T
  - Fonde caratteristiche di SunOS, 4.3BSD, Microsoft Xenix
- SVR4 è conforme a POSIX e XPG3



## Implementazioni: 4.3,4.4BSD

- Berkeley Software Distributions ( sono distribuite da UCB)
- Conforme allo standard POSIX
- Benchè fosse inizialmente legato a codice sorgente AT&T e quindi alle sue licenze, è stata creata una versione (free) molto interessante per PC (intel based) FreeBSD

## Implementazioni: Linux

- Il primo kernel sviluppato da Linus Torvalds nel 1991
- Conforme a POSIX, ma include anche la maggior parte di funzioni di SVR4 e 4.3BSD
- Disponibile su Intel, Compaq (ex Digital), Alpha, Sparc, Mac e Amiga
- Free



## Distribuzioni Linux

- Slackware
- Debian
- RedHat
- SuSe
- Mandrake



## Limiti

- Numeri magici e costanti definiti nelle varie implementazioni
- Tipi di limiti:
  - Opzioni e limiti in fase di compilazione
    - Possono essere definiti in headers che un qualunque programma può includere a tempo di compilazione
  - Limiti run-time
    - Prevede che il processo chiami una funzione per ottenere il valore del limite
    - Limiti non associati a file/dir → funzione sysconf
    - Limiti associati a file/dir → funzione pathconf
- Ansi C: solo limiti a tempo di compilazione (cf. <limits.h>)
- POSIX e XSI: entrambi i tipi, limiti indeterm.



## Tipi di dati primitivi

- Nel file header <sys/types.h> (come anche in altri header) sono definiti alcuni tipi di dati system-dependent chiamati tipi di dati primitivi
  - Sono definiti utilizzando typedef
  - I loro nomi finiscono in genere con _t
- La tabella 2.20 mostra i principali tipi di dati primitivi che useremo
- Un primo esercizietto:
  - Trovare dove sono definiti in Linux