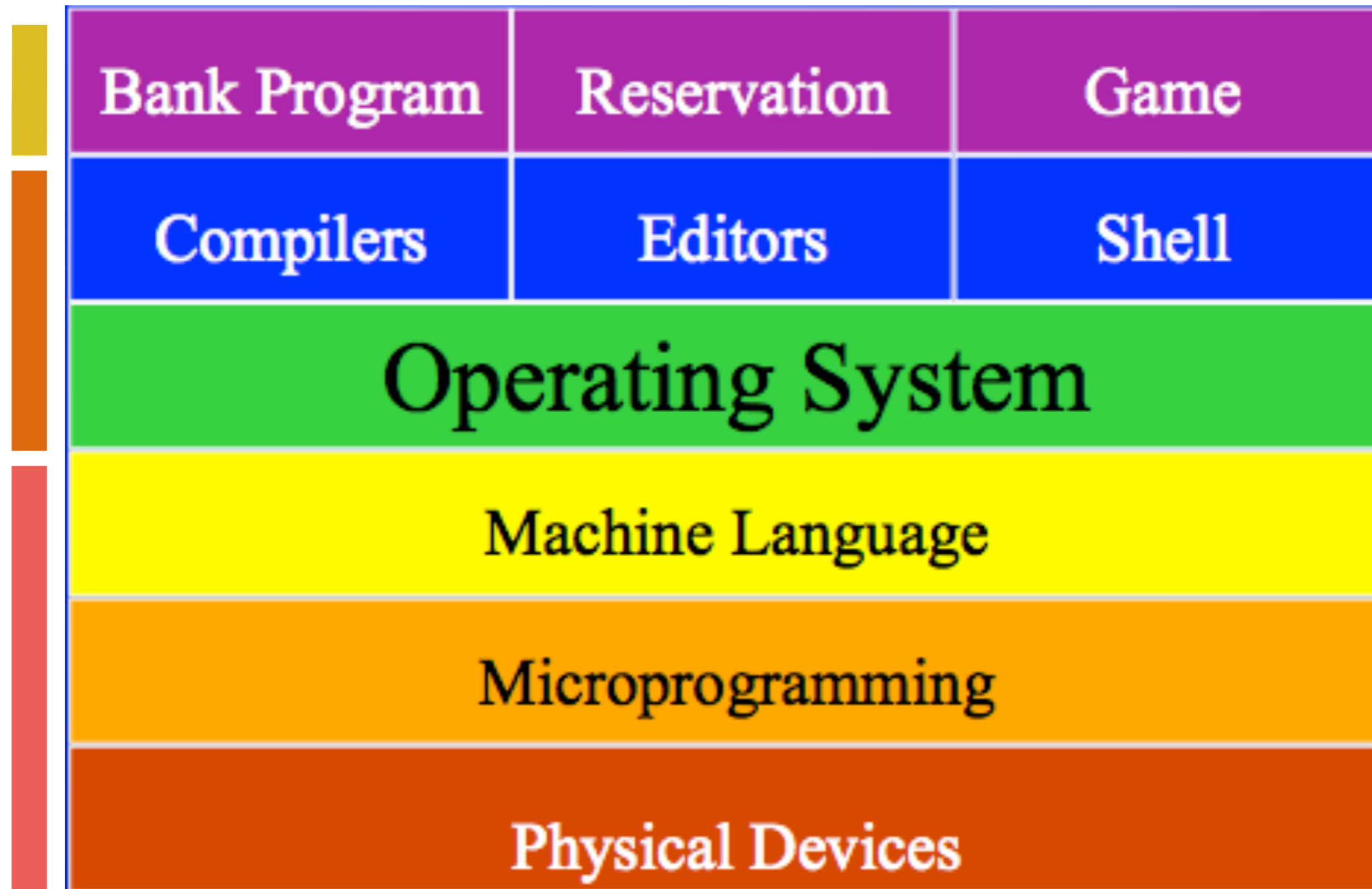


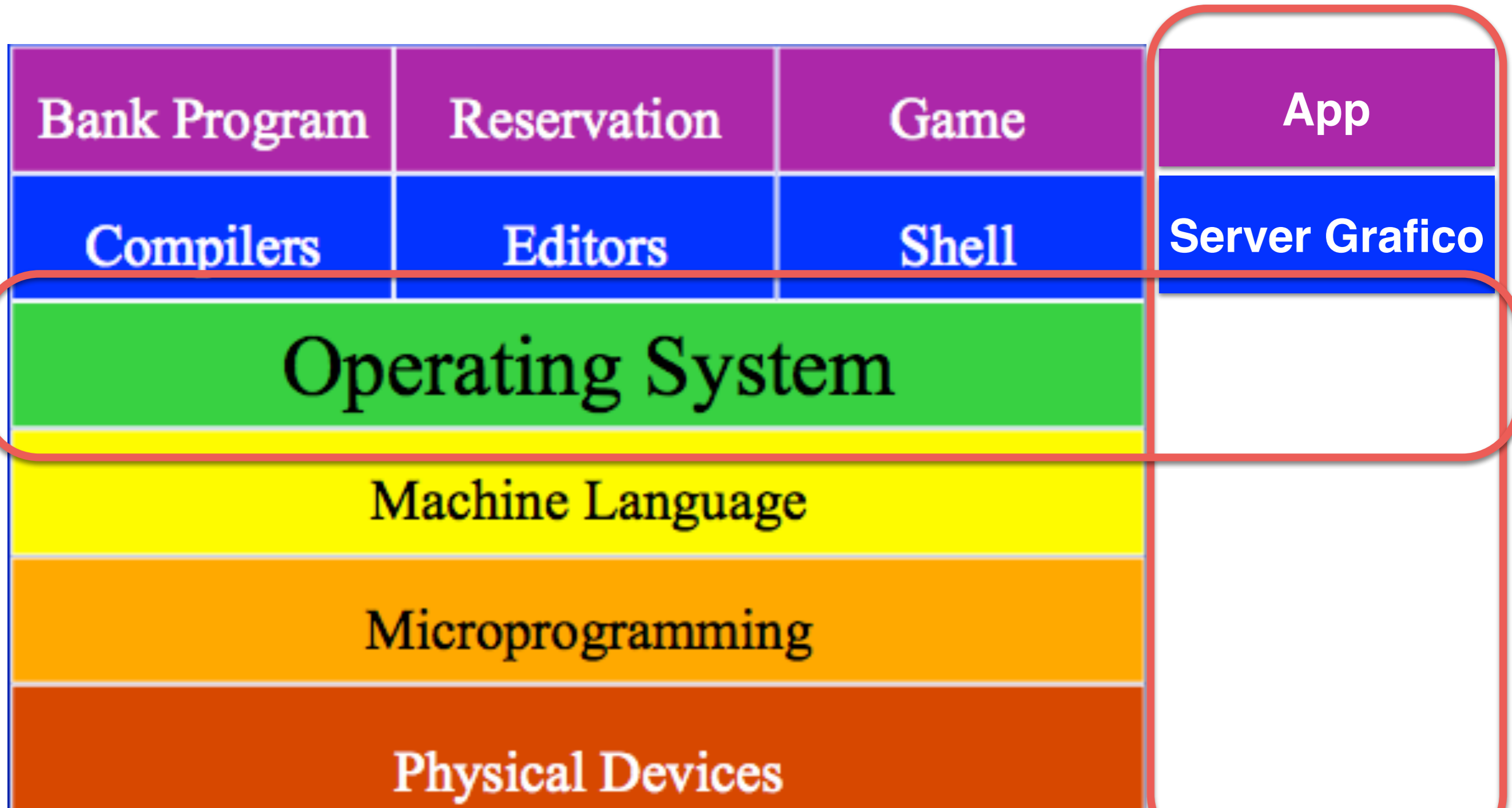
# UNIX Basics

Franco Maria Nardini

# Sistema Operativo I



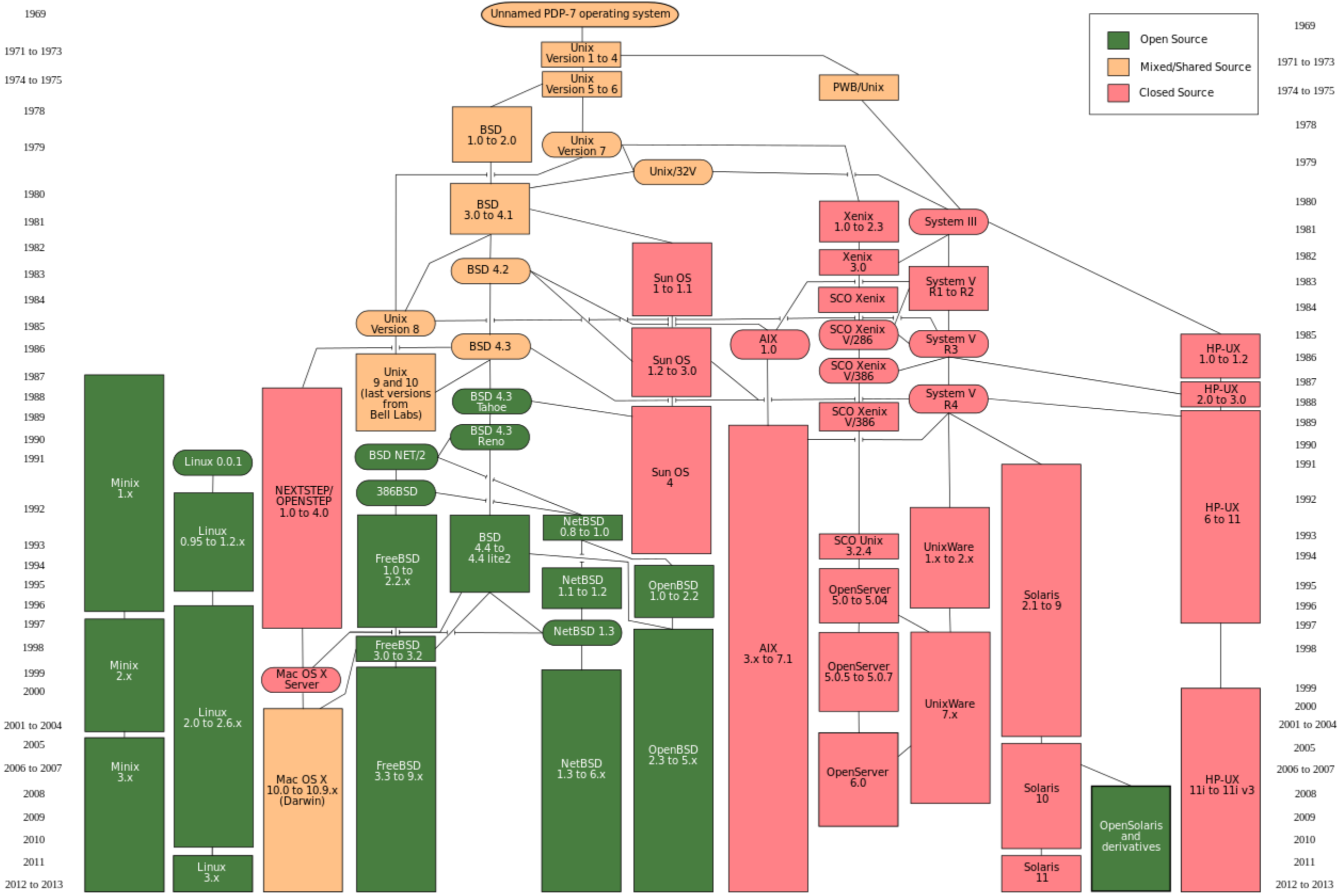
# Sistema Operativo II



# UNIX

- scritto ai Bell Labs (New Jersey) dal 1969 da Ken Thompson and Dennis Ritchie.
- Dal 1973 totalmente in C.
- Nel 1974 Thompson, Joy, Haley e studenti sviluppano il “Berkeley Software Distribution” (BSD) of UNIX
- Evolve in una “galassia” di versioni:
  - due direzioni: BSD e “System V”





■	Open Source
■	Mixed/Shared Source
■	Closed Source

1969  
1971 to 1973  
1974 to 1975  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001 to 2004  
2005  
2006 to 2007  
2008  
2009  
2010  
2011  
2012 to 2013

1969  
1971 to 1973  
1974 to 1975  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001 to 2004  
2005  
2006 to 2007  
2008  
2009  
2010  
2011  
2012 to 2013

# Alcune Date Importanti

- 1984: BSD 4.2 rilascia TCP/IP
- 1986: BSD 4.3 rilascia NFS
- 1991: Linus Torvalds inizia a lavorare al kernel Linux
- 1994: unica specifica UNIX introdotta
- 2000: Darwin (derivato da NeXT, FreeBSD, NetBSD)
- 2007: iOS
- 2008: Android

# Alcune Versioni :)

1BSD	2BSD	3BSD	4BSD	4.4BSD Lite 1
4.4BSD Lite 2	386 BSD	A/UX	Acorn RISC iX	AIX
AIX PS/2	AIX/370	AIX/6000	AIX/ESA	AIX/RT
AMiX	AOS Lite	AOS Reno	ArchBSD	ASV
Atari Unix	BOS	BRL Unix	BSD Net/1	BSD Net/2
BSD/386	BSD/OS	CB Unix	Chorus	Chorus/MiX
Coherent	CTIX	Darwin	Debian GNU/Hurd	DEC OSF/1 ACP
Digital Unix	DragonFly BSD	Dynix	Dynix/ptx	ekkoBSD
FreeBSD	GNU	GNU-Darwin	HPBSD	HP-UX
HP-UX BLS	IBM AOS	IBM IX/370	Interactive 386/ix	Interactive IS
IRIX	Linux	Lites	LSX	Mac OS X
Mac OS X Server	Mach	MERT	MicroBSD	Mini Unix
Minix	Minix-VMD	MIPS OS	MirBSD	Mk Linux
Monterey	more/BSD	mt Xinu	MVS/ESA OpenEdition	NetBSD
NeXTSTEP	NonStop-UX	Open Desktop	Open UNIX	OpenBSD
OpenServer	OPENSTEP	OS/390 OpenEdition	OS/390 Unix	OSF/1
PC/IX	Plan 9	PWB	PWB/UNIX	QNX
QNX RTOS	QNX/Neutrino	QUNIX	ReliantUnix	Rhapsody
RISC iX	RT	SCO UNIX	SCO UnixWare	SCO Xenix
SCO Xenix System V/386	Security-Enhanced Linux	Senix	Senix ReliantUnix	Solaris
SPIX	SunOS	Tru64 Unix	Trusted IRIX/B	Trusted Solaris
Trusted Xenix	TS	UCLA Locus	UCLA Secure Unix	Ultrix
Ultrix 32M	Ultrix-11	Unicos	Unicos/mk	Unicox-max
UNICS	UNIX 32V	UNIX Interactive	UNIX System III	UNIX System IV
UNIX System V	UNIX System V Release 2	UNIX System V Release 3	UNIX System V Release 4	UNIX System V/286
UNIX System V/386	UNIX Time-Sharing System	UnixWare	UNSW	USG
Venix	Wollogong	Xenix OS	Xinu	xMach

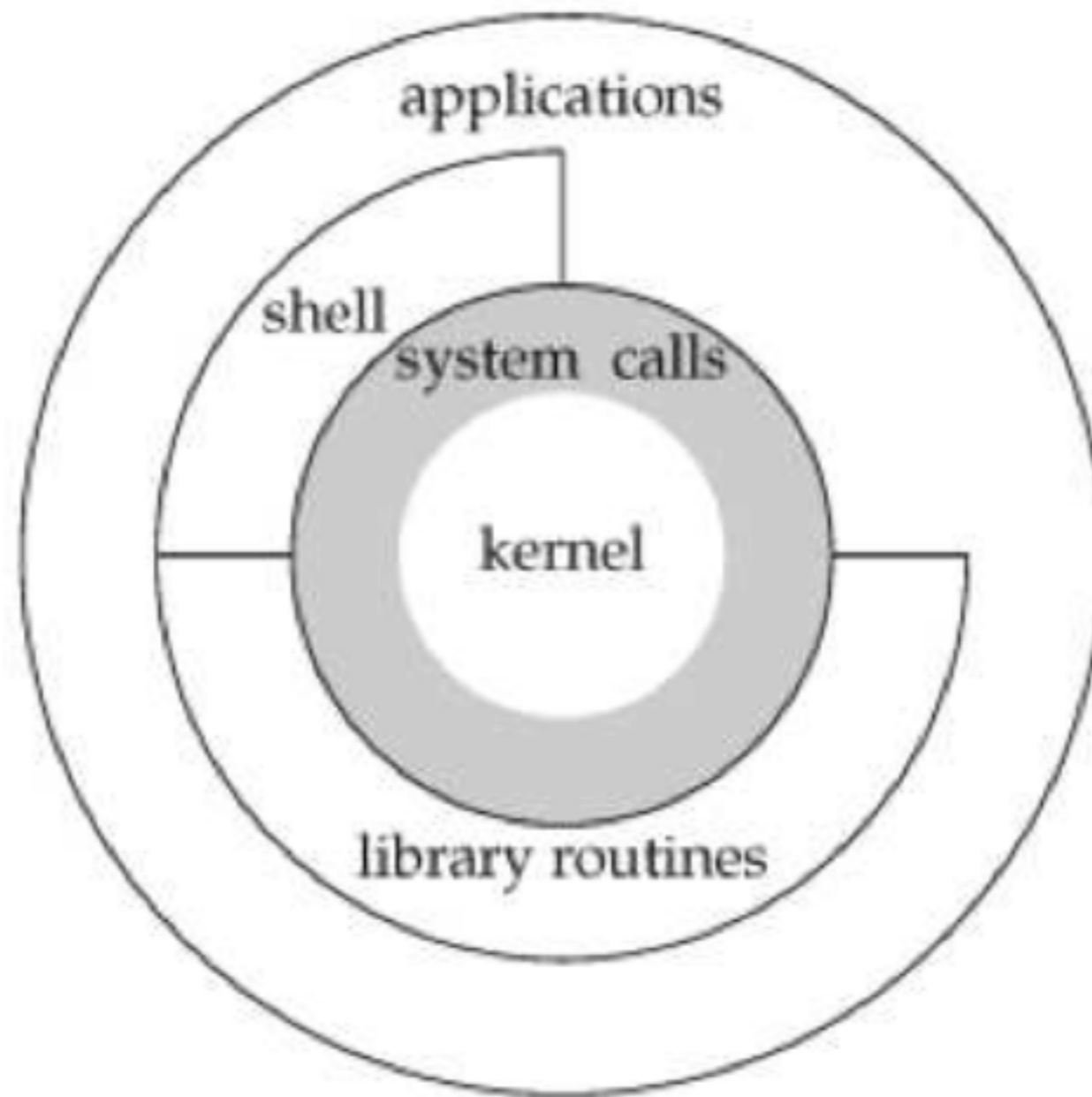
# UNIX dappertutto

sistemi desktop, server, cloud, TV, telefoni, orologi,  
stereo, navigatori satellitari, termostati, sensori,  
sistemi embedded, serrature intelligenti,  
player musicali

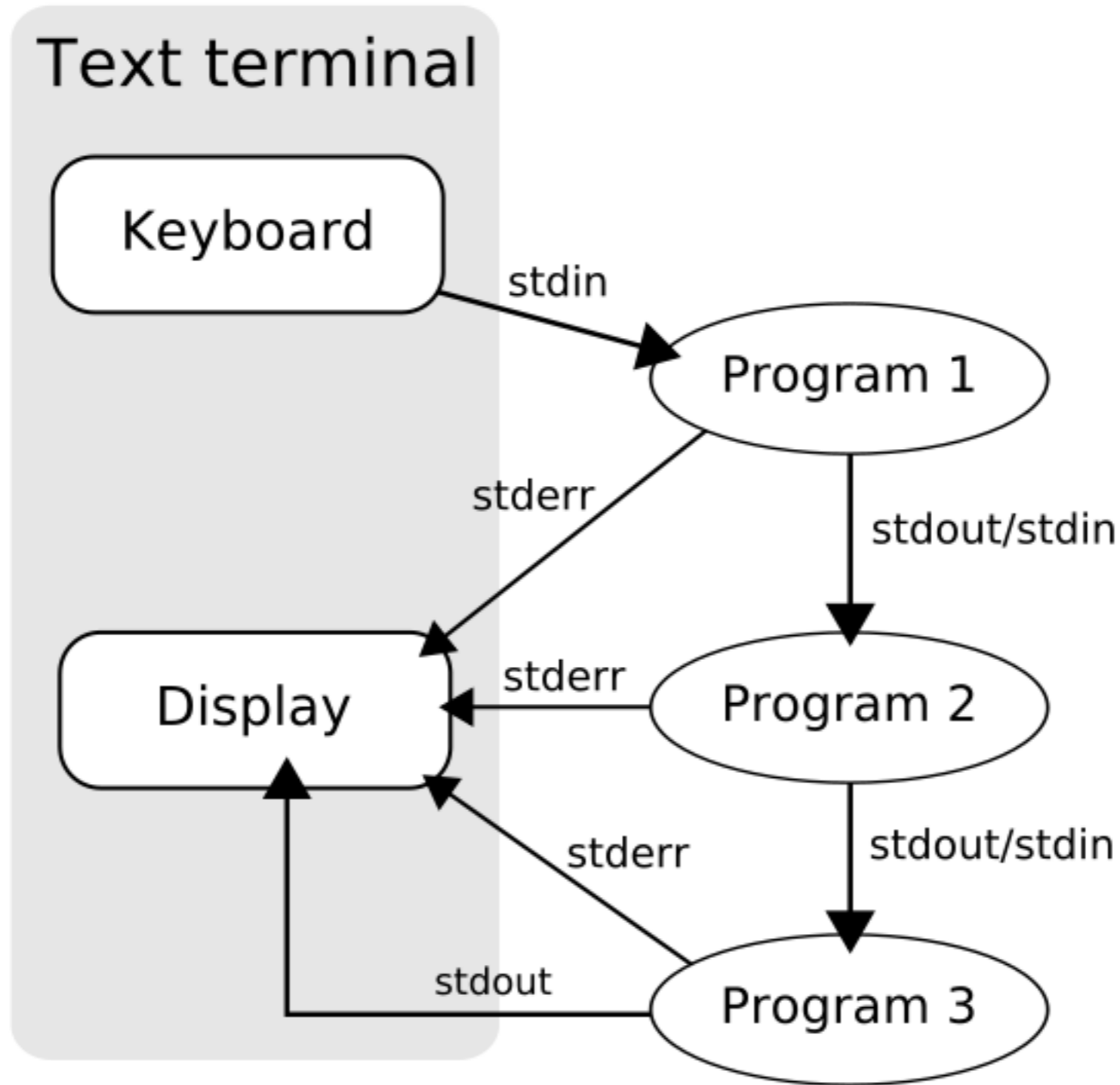
funzionano spesso grazie a sistemi UNIX o derivati!



# Concetti Base I



# Concetti Base II



# Ispirazione

“Consistency underlies all principles of quality.”  
Frederick P. Brooks, Jr

# Filosofia dietro UNIX

- un mondo: [https://en.wikipedia.org/wiki/Unix\\_philosophy](https://en.wikipedia.org/wiki/Unix_philosophy)

- I programmi UNIX sono:

- semplici

- seguono la regola della “più piccola sorpresa”

- input su `stdin`, output su `stdout`, errori su `stderr`

- usano gli `exit codes`

- hanno una pagina `man`



# Pipelining I

**Problema:** avete un file di 100 numeri interi positivi, uno per riga e volete ordinarlo dal più piccolo, al più grande

**Soluzione?** (5 min)

**Si può fare meglio? :)**

**Potenza del pipelining e della filosofia UNIX**

```
cat primi100numeri.txt | sort -n
```

# Pipelining II

**Problema:** avete un file di 100 numeri interi positivi, uno per riga e volete sapere quante occorrenze ci sono di ogni numero

**Soluzione?** (5 min)

**Si può fare meglio? :)**

**Potenza del pipelining e della filosofia UNIX**

```
cat primi100numeri.txt | sort -n | uniq -c
```

# Boot

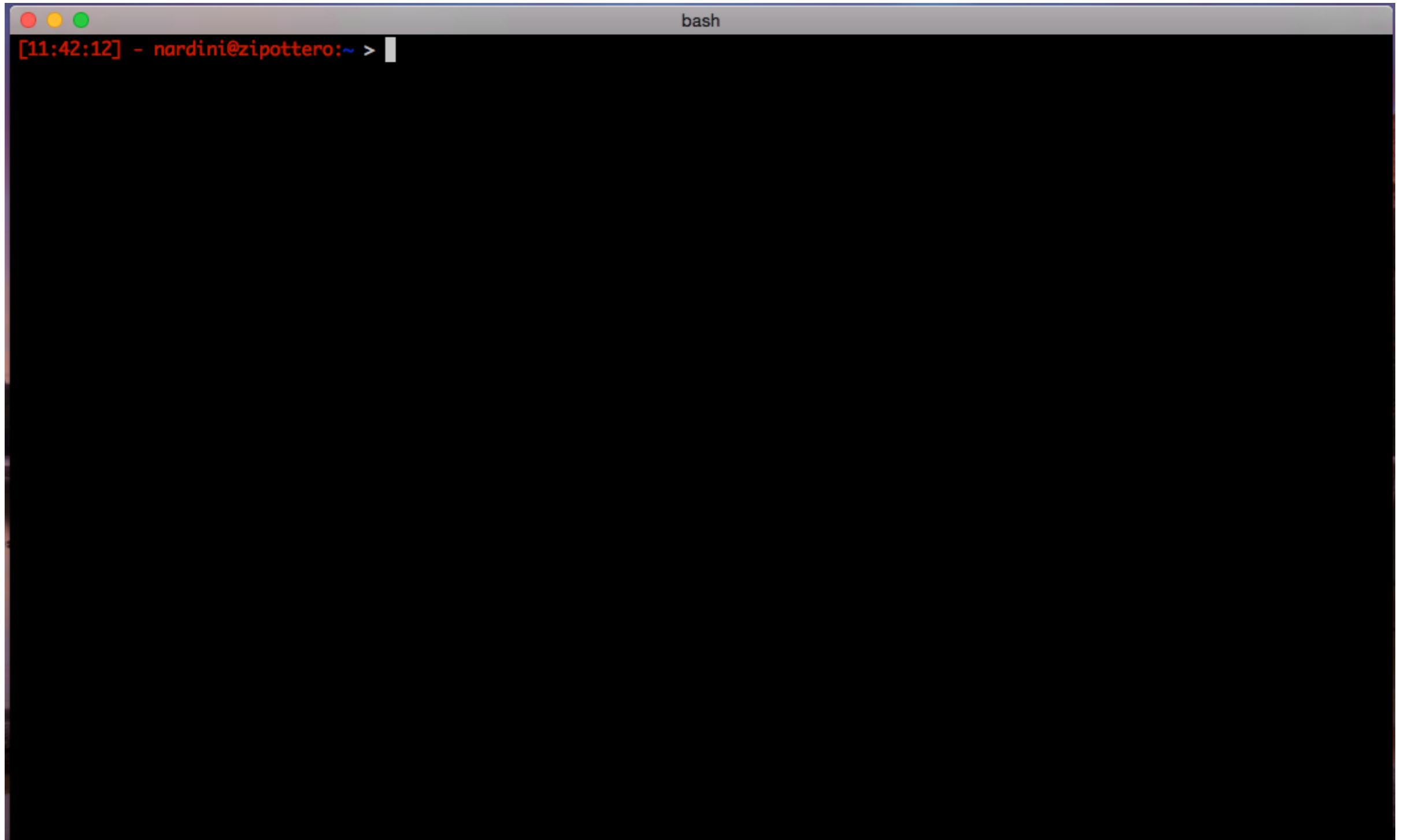
```
[...]  
total memory = 768 MB  
avail memory = 732 MB  
timecounter: Timecounters tick every 10.000 mse  
mainbus0 (root)  
[...]  
boot device: xbd3  
root on xbd3a dumps on xbd3b  
mountroot: trying lfs...  
mountroot: trying ffs...  
root file system type: ffs  
init: copying out path '/sbin/init' 11  
[...]  
Starting local daemons:.  
Starting sendmail.  
Starting sshd.  
Starting snmpd.  
Starting cron.  
  
NetBSD/amd64 (panix.netmeister.org) (console)  
  
login:
```

# Boot

```
[...]  
total memory = 768 MB  
avail memory = 732 MB  
timecounter: Timecounters tick every 10.000 msec  
mainbus0 (root)  
[...]  
boot device: xbd3  
root on xbd3a dumps on xbd3b  
mountroot: trying lfs...  
mountroot: trying ffs...  
root file system type: ffs  
init: copying out path '/sbin/init' 11  
[...]  
Starting local daemons:.  
Starting sendmail.  
Starting sshd.  
Starting snmpd.  
Starting cron.  
  
NetBSD/amd64 (panix.netmeister.org) (console)  
  
login:  
Password:  
Last login: Sat Sep 10 14:27:56 2011 on console  
Copyright (c) 1982, 1986, 1989, 1991, 1993  
The Regents of the University of California. All rights reserved.  
  
NetBSD 5.0.2 (PANIX-VC) #2: Tue Oct 19 16:30:57 EDT 2010  
  
Welcome to NetBSD!  
  
$
```



# Command Line

A screenshot of a terminal window. The window title bar shows three colored window control buttons (red, yellow, green) on the left and the text "bash" on the right. The terminal content shows a red prompt string "[11:42:12] - nardini@zipottero:~ >" followed by a white cursor bar. The rest of the terminal area is black and empty.

```
[11:42:12] - nardini@zipottero:~ > |
```

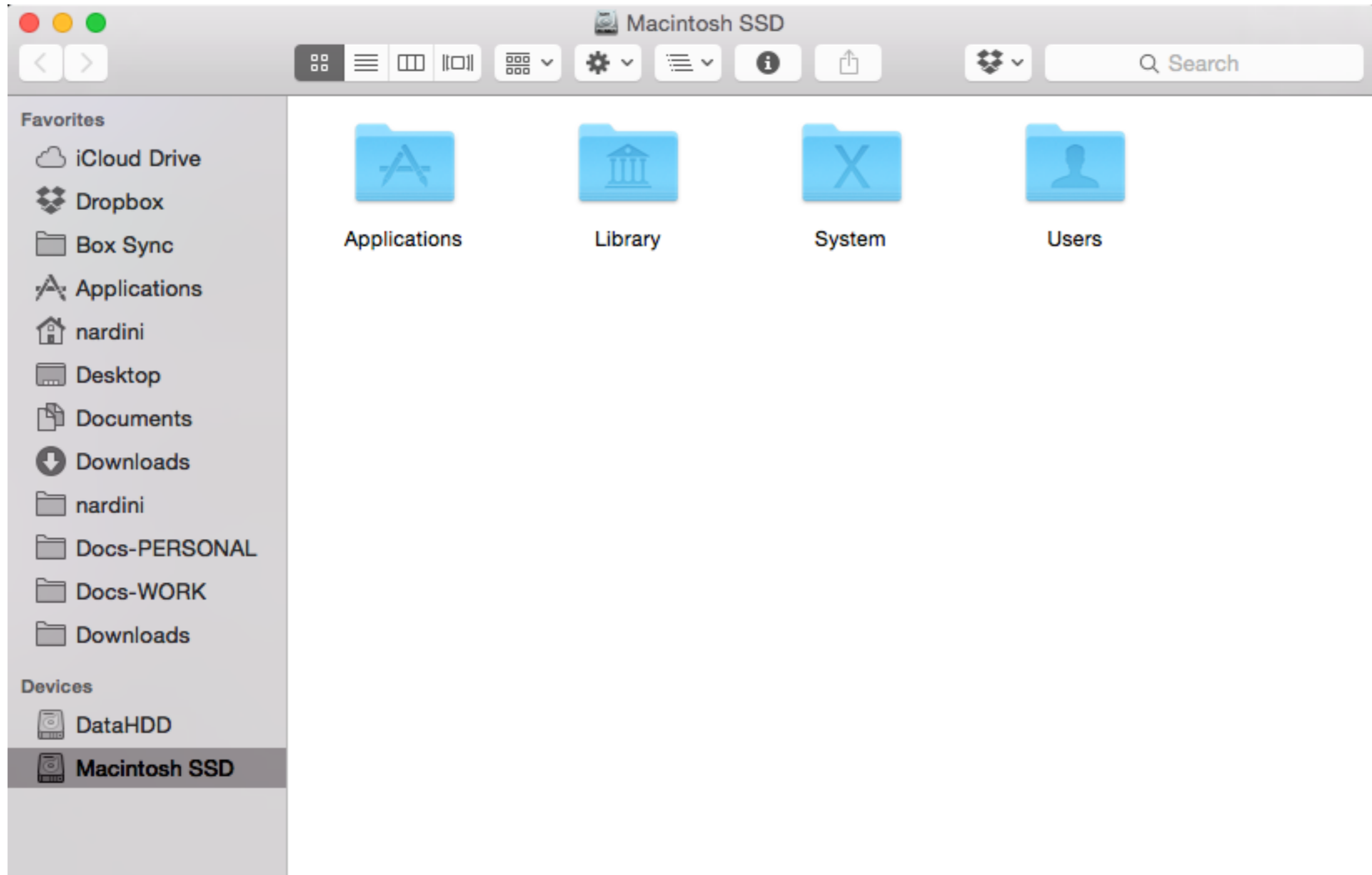
# Filesystem

- Il filesystem UNIX e' organizzato ad **albero**
- la radice del filesystem è: /
- le *partizioni* sono “montate” sotto la radice
- Un nome di file può essere qualsiasi carattere ad eccezione di **NUL** e /
- il path di un file può essere una sequenza di nomi (come sopra) intervallati da /

# Filesystem

```
[10:58:36] - nardini@zipottero:~ > ls -l /
total 45
drwxrwxr-x+ 71 root  admin  2482 Mar 10 10:08 Applications
drwxr-xr-x+ 67 root  wheel  2346 Oct 31 16:24 Library
drwxr-xr-x@  2 root  wheel   68 Sep  9 2014 Network
drwxr-xr-x+  3 root  wheel  136 Oct 31 11:38 System
drwxr-xr-x   5 root  admin  204 Feb 25 16:59 Users
drwxrwxrwt@  3 root  admin  136 Mar 14 09:00 Volumes
drwxr-xr-x@  2 root  wheel 1326 Jan 30 18:15 bin
drwxrwxr-t@  2 root  admin   68 Sep  9 2014 cores
dr-xr-xr-x   3 root  wheel 4351 Mar 14 09:00 dev
lrwxr-xr-x@  1 root  wheel  11 Oct 31 11:33 etc -> private/etc
dr-xr-xr-x   2 root  wheel   1 Mar 14 10:32 home
-rw-r--r--@  1 root  wheel  313 Oct  1 08:12 installer.failurerequests
dr-xr-xr-x   2 root  wheel   1 Mar 14 10:32 net
drwxr-xr-x@  3 root  wheel  102 Aug 11 2014 opt
drwxr-xr-x@  6 root  wheel  204 Oct 31 11:40 private
drwxr-xr-x@  2 root  wheel 2006 Jan 30 18:15 sbin
lrwxr-xr-x@  1 root  wheel  11 Oct 31 11:35 tmp -> private/tmp
drwxr-xr-x@ 10 root  wheel  374 Oct 31 16:24 usr
lrwxr-xr-x@  1 root  wheel  11 Oct 31 11:35 var -> private/var
[10:58:39] - nardini@zipottero:~ > █
```

# Filesystem



# Filesystem

- Ogni cosa è un file!
- le directory sono file speciali che contengono il mapping tra `inodes` e i nomi di file
- Un processo ha una “working directory” da cui i path relativi sono specificati
  - il path è relativo se inizia dalla working directory
  - il path è assoluto se inizia da root

# ls

```
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {

    DIR *dp;
    struct dirent *dirp;

    if (argc != 2) {
        fprintf(stderr, "usage: %s dir_name\n", argv[0]);
        exit(1);
    }

    if ((dp = opendir(argv[1])) == NULL ) {
        fprintf(stderr, "can't open '%s'\n", argv[1]);
        exit(1);
    }

    while ((dirp = readdir(dp)) != NULL )
        printf("%s\n", dirp->d_name);

    closedir(dp);
    return(0);
}
```

# Esempi

```
$ wget http://hpc.isti.cnr.it/~nardini/siselab/01/simple-ls.c
```

```
$ more simple-ls.c
```

```
$ cc -Wall -o myls simple-ls.c
```

```
$ ./mysls .
```

(esempio pratico)

# Utenti e Gruppi

- Utenti (`root`) e gruppi sono rappresentati con identificatori: `userids` e `groupids`
- Essi sono utilizzati per identificare utenti e garantire i giusti “permessi” nel sistema.
- `groupids` sono di due tipi: `primary`, `secondary`

\$ `id`

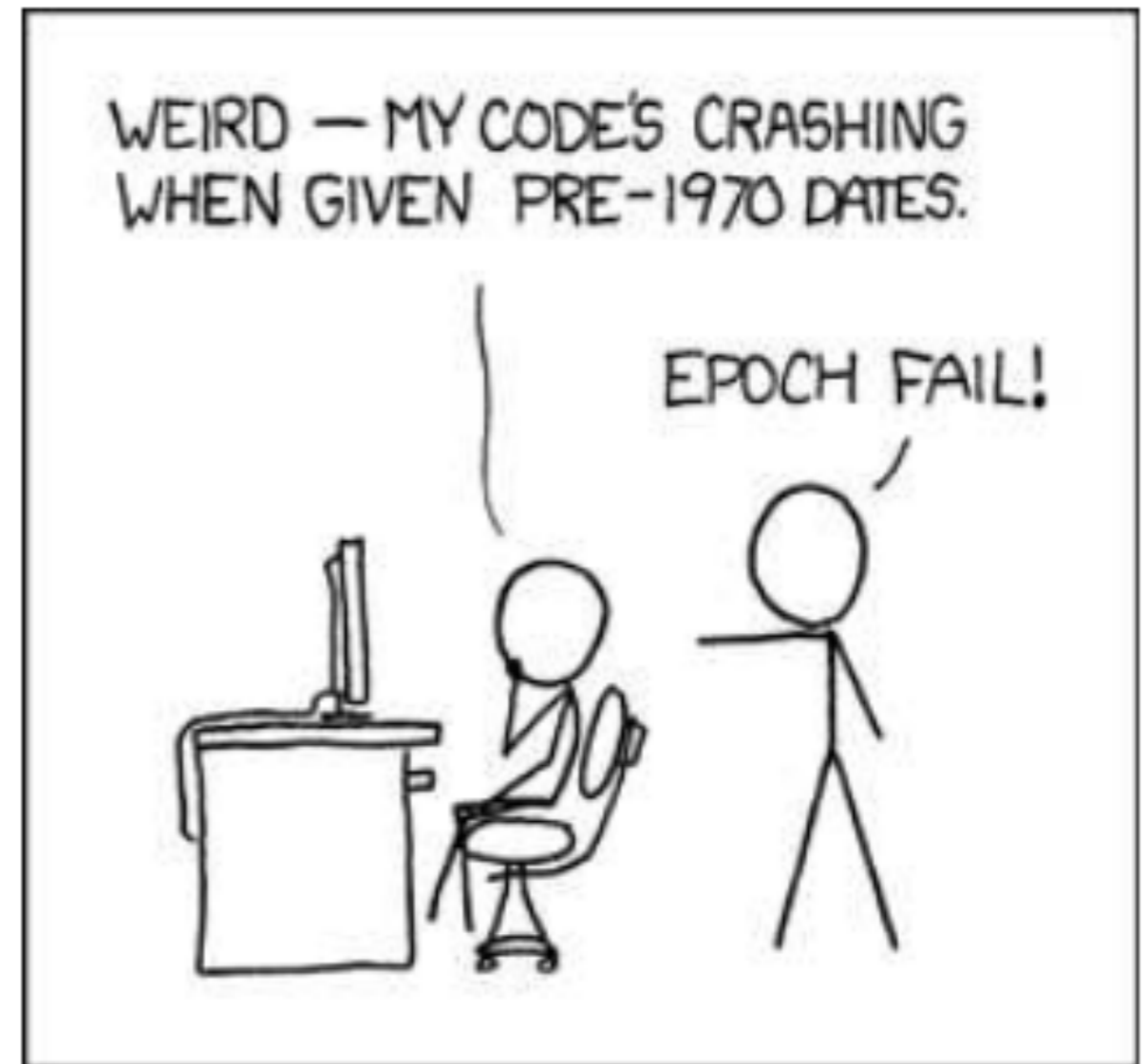


# UNIX Time

- Calendar Time: misurato in secondi dall'inizio dell'epoca UNIX (01 gennaio 1970)
- memorizzato con un tipo di variabile: `time_t`

```
$ date +%s
```

- [https://en.wikipedia.org/wiki/Year\\_2038\\_problem](https://en.wikipedia.org/wiki/Year_2038_problem)



# Valori di UNIX Time

- Process time: risorse CPU usate dal processo. Misurate in *clock ticks* (`clock_t`). Tre valori
  - clock time
  - user CPU time
  - system CPU time

```
$ time ls
```

```
$ time grep -r _POSIX_SOURCE /usr/include > /dev/null
```

# System e Library Calls

- Una *system call* è una entry point al kernel dove la specifica funzione richiesta è implementata
  - documentate in sez. 2 del manuale (`write(2)`)
- Una *library call* è una chiamata a codice utente che implementa una specifica operazione
  - documentate in sez. 3 del manuale (`printf(3)`)

# Standards

- Linguaggio C
  - ANSI C (X3.159-1989) C89
  - C9X/C99 (ISO/IEC 9899)
  - C11 (ISO/IEC 9899:2011)
- Portable Operating System Interface (POSIX)
  - IEEE Computer Society
  - definisce compatibilità tra sistemi operativi
  - application programming interface (API), command line shells and utility interfaces

# Esempi

```
$ wget http://hpc.isti.cnr.it/~nardini/siselab/01/simple-shell.c
```

```
$ more simple-shell.c
```

```
$ cc -Wall -o mysh simple-shell.c
```

```
$ ./mysh .
```