Corso «Sistemi Operativi» AA 2019-2020

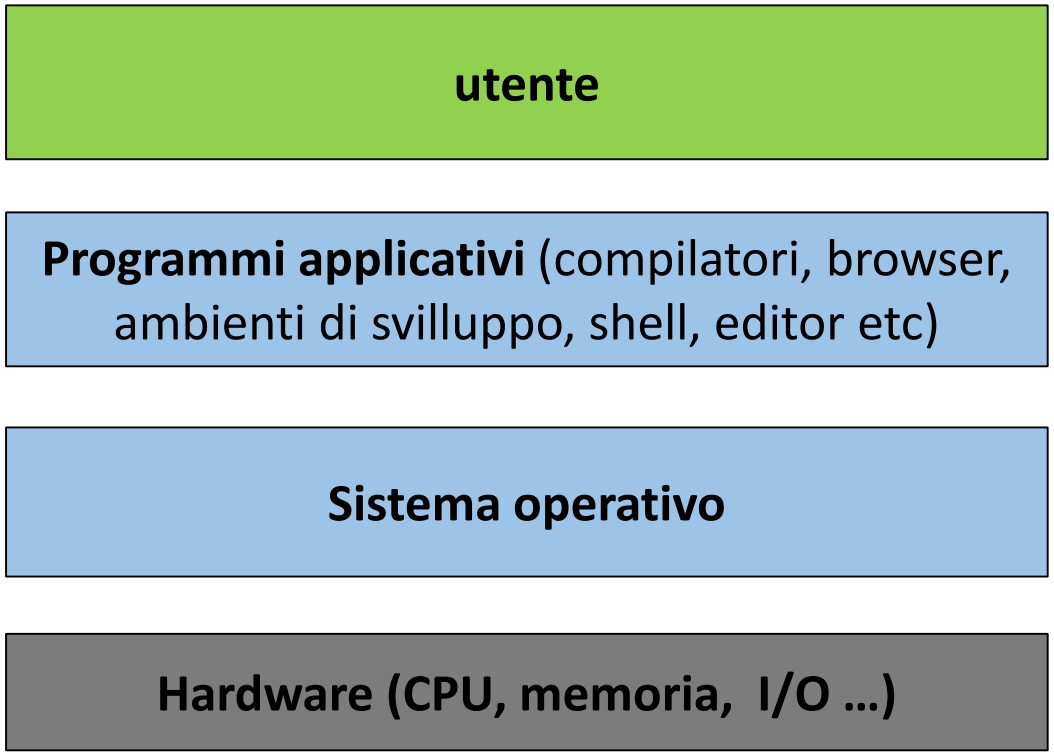
Marco Tessarotto

Cosa è un Sistema Operativo?

Sistema operativo: insieme di programmi (software) che gestisce gli elementi fisici di un calcolatore (hardware).

Componenti di un calcolatore

- L'utente utilizza il computer attraverso l'interfaccia utente (tastiera, mouse, schermo, touchscreen, riconoscimento vocale…)
- I programmi applicativi utilizzano le risorse di elaborazione per fornire strumenti agli utenti per risolvere problemi
- Il sistema operativo controlla l'hardware e ne coordina l'utilizzo da parte dei programmi applicativi
- L'hardware fornisce le risorse di elaborazione



Caratteristiche di un SO

- Il sistema operativo gestisce ed **assegna le risorse** del calcolatore ai programmi
- Tempo di CPU, spazio di memoria, spazio per la memorizzazione dei file, dispositivi di I/O
- Il SO agisce come **gestore** di tali **risorse**.
- Di fronte a richieste multiple e/o conflittuali di risorse, il SO decide come assegnarle agli specifici programmi ed utenti affinché il computer operi in modo equo ed efficiente

Breve storia di UNIX

Breve storia di UNIX

- Linux è un SO che fa parte della famiglia dei SO UNIX

- La prima implementazione di un SO UNIX fu realizzata nel 1969: Ken Thompson ai Bell Laboratories, divisione di AT&T.
- Fu scritto per il minicomputer Digital PDP-7

1965: 500 Kg, 4K words

Linguaggio di programmazione C

- Dennis Ritchie, collega di Thompson, cominciò a sviluppare il linguaggio C e nel 1973 il SO UNIX fu riscritto quasi completamente in linguaggio C
- UNIX divenne uno dei primo OS scritti in un linguaggio ad «alto livello» => quindi «facilmente» portabile su altre architetture hardware

- Il linguaggio C nasce fin dall'inizio per scrivere SO

Storia di UNIX

- Dal 1969 al 1979, UNIX fu sviluppato attraverso varie “release” (rilasci) :
- First edition, 1971
- Second edition, 1972
- Third edition, 1973
- Fourth edition, 1973
- Fifth edition, 1974: UNIX installato su più di 50 sistemi
- Sixth edition, 1975: usato fuori da AT&T

BSD e System V

- Fino a questo punto: UNIX è di proprietà di AT&T che non può venderlo in quando monopolista nel campo telefonia fissa.
- Ma, dal 1974, AT&T concede l'uso di UNIX alle università USA… che ci mettono mano e contribuiscono allo sviluppo di UNIX. Lo possono fare perché AT&T fornisce, con la licenza, anche il **codice sorgente** di UNIX.
- Seventh edition, 1979

- Da questo punto UNIX diverge in: BSD e System V

BSD

- Thompson nel 1975/76 è professore a Berkeley, Università della California: insegna UNIX
- Nasce la **Berkeley Software Distribution** (BSD) con molti miglioramenti, nuovi strumenti etc.

- La prima distribuzione è 3BSD nel dicembre 1979

BSD e … Internet

- 4.2BSD nel 1983, contiene una implementazione completa di **TCP/IP**
- 4.3BSD nel 1986
- 4.4BSD nel 1993

- 4.2BSD viene distribuita in molte università nel mondo

System V

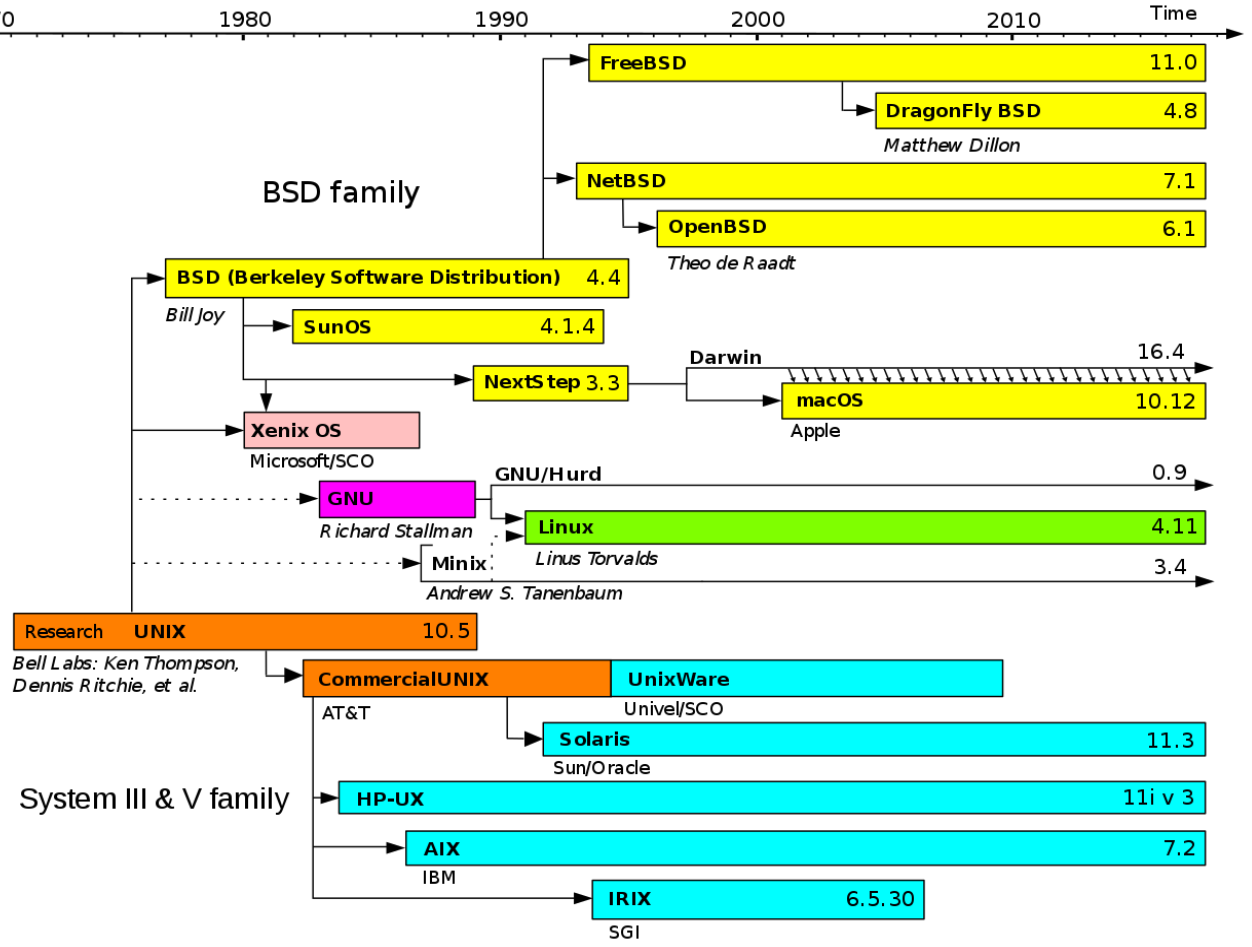
- Nel frattempo: AT&T viene “spezzettata” dal governo USA nel 1982… AT&T può vendere licenze commerciali di UNIX :
- Release System III, 1981
- System V, 1983

- System V Release 4, 1989

Microsoft (ed altri) e UNIX

System III è stato utilizzato (su licenza commerciale) da aziende terze per creare altri SO proprietari derivati da UNIX:

SunOS (di Sun Microsystems), Apple Macintosh, SCO XENIX (Microsoft)



Free Software Foundation

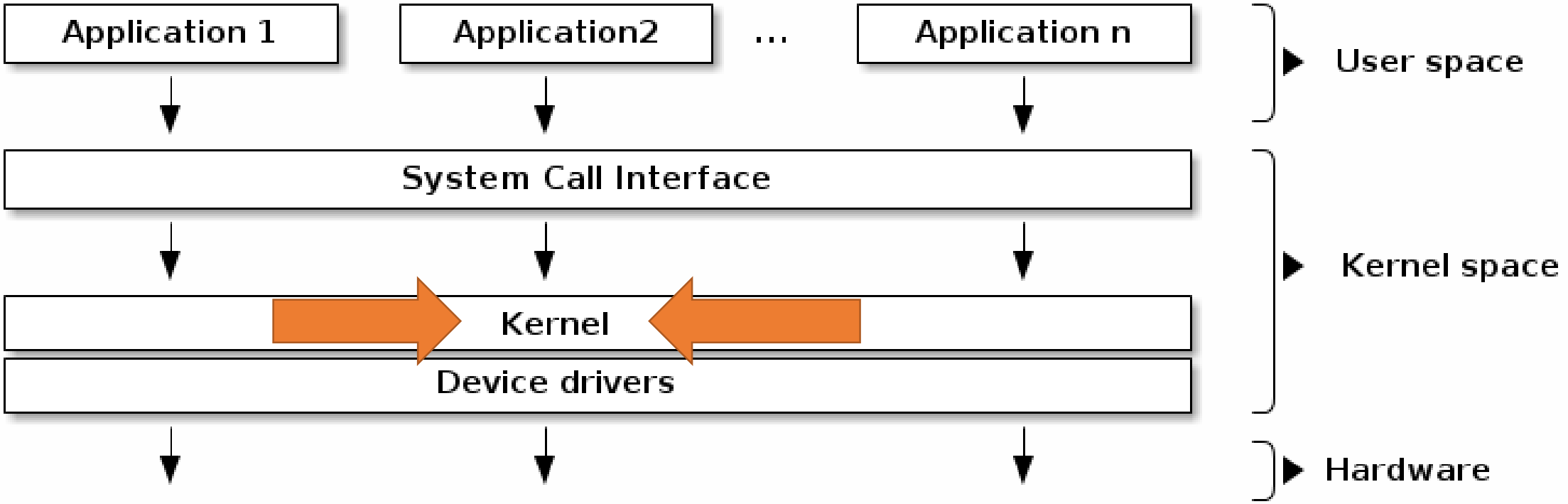
- Nel 1984, Richard Stallman iniziò a lavorare su una versione “free” di UNIX (libera da diritti commerciali, patenti…) e nel 1985 fonda la Free Software Foundation (FSF), organizzazione no-profit.
- Inizia il progetto GNU (GNU is not UNIX). Viene sviluppata (1989) la licenza GNU General Public License (**GPL**). Software licenziato con la GPL deve essere reso disponibile nel suo codice sorgente e deve essere liberamente distribuibile.

Progetto GNU

- Il Progetto GNU ha prodotto molti programmi (ma **non** un SO UNIX completo: mancava il «**kernel**») **??? Kernel ???**

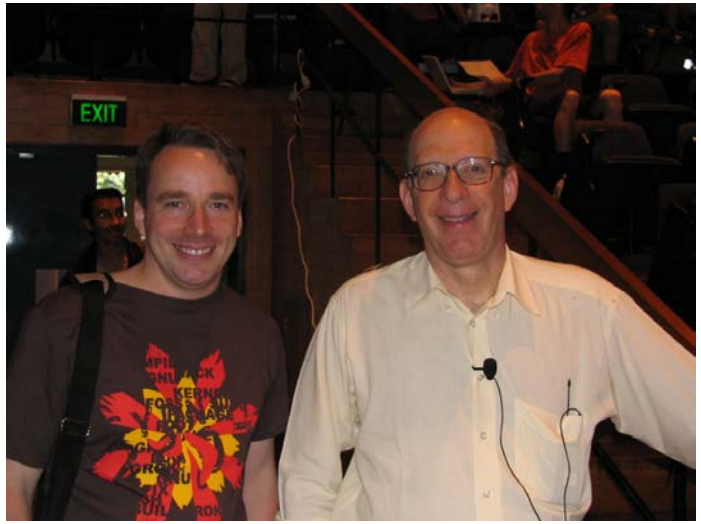
- Il progetto GNU ha prodotto: emacs, gcc (GNU C compiler), bash shell, glibc (the GNU C library)

Architettura tipica del sistema operativo



Linus Torvalds

- Nel 1991, Linus Torvalds è uno studente finlandese presso l'Università di Helsinki
- Nel corso degli studi, era entrato in contatto con Minix, un SO Unix-like, sviluppato a scopo didattico dal prof. Andrew Tanenbaum (ma Minix non era molto efficiente in quanto era, per prima cosa, uno strumento didattico)
- Torvalds iniziò a scrivere per hobby un SO per il suo PC basato su Intel 80386

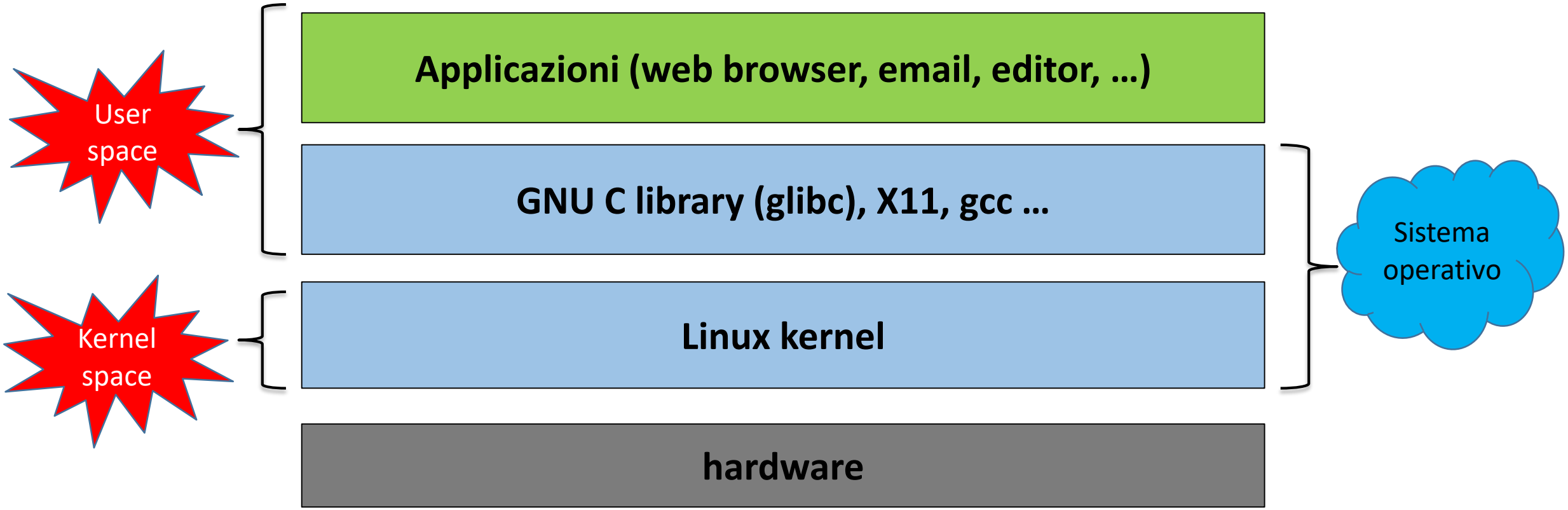


Linux Kernel

- Quindi, Torvalds decise di iniziare a scrivere il suo UNIX kernel (1991)…

- Il 5 ottobre 1991 pubblicò su Usenet un messaggio, pubblicando la versione 0.02 del Kernel Linux (quasi subito con licenza GPL). Altri programmatori si unirono a Torvalds nello sviluppo del kernel Linux

OS Linux


User space
Kernel space
Applicazioni (web browser, email, editor, …)
GNU C library (glibc), X11, gcc …
Linux kernel
hardware
Sistema operativo

Linux Kernel

- 1991: avvio del progetto
- Marzo 1994: Linux Kernel versione 1.0
- Marzo 1995: Linux Kernel 1.2
- Giugno 1996: Linux Kernel 2.0
- Gennaio 1999: Linux Kernel 2.2
- Gennaio 2001: Linux Kernel 2.4
- Dicembre 2003: linux Kernel 2.6
- …
- Gennaio 2020: Linux Kernel 5.5 (https://www.kernel.org oppure github.com)

BSD – un altro UNIX «libero»

- Nei primi anni 90, era disponibile un altro UNIX free per intel x86-32: il porting di BSD per x86-32, conosciuto come 386/BSD. Era basato sul source code di 4.3BSD (che non conteneva praticamente più codice di AT&T).
- 386/BSD si divise in due progetti paralleli: NetBSD (portabilità) e FreeBSD (performance).
- NetBSD 0.8, aprile 1993
- FreeBSD 1.0, dicembre 1993
- OpenBSD (security), 1996, fork di NetBSD

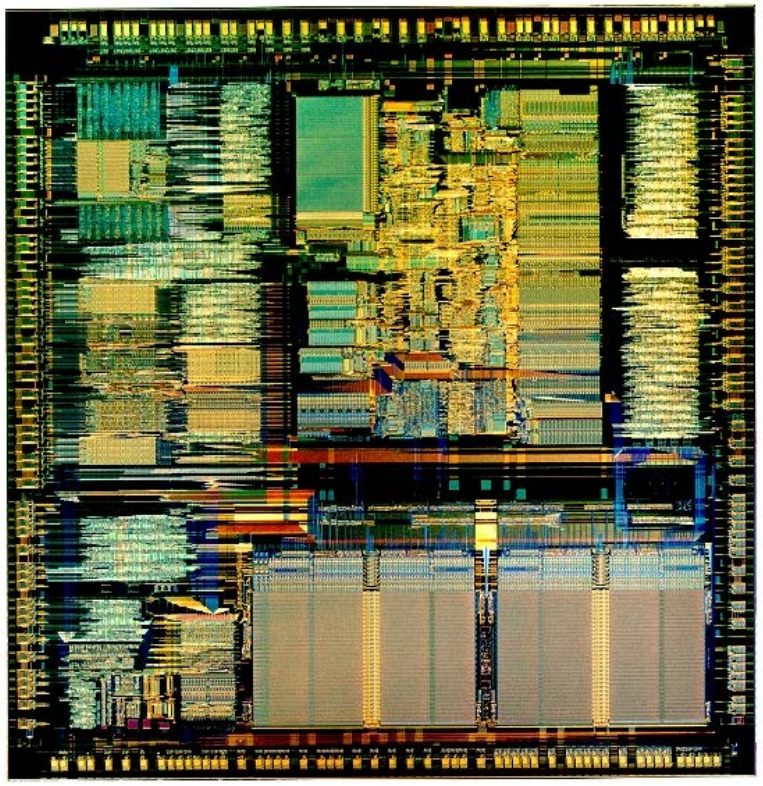
- Nel 1992 AT&T iniziò una causa verso Berkeley… fino al 1994
- risultato: 4.4BSD-Lite, Release 2, giugno 1995

Linux ed altre architetture

- All’inizio dell sviluppo di Linux, l’obiettivo principale era lo sviluppo di una implementazione efficiente per il solo processore Intel 80386

- Con il crescere della popolarità di Linux, fu portato su altre architetture quali: **x86-64**, Motorola/IBM PowerPC, Sun SPARC, MIPS, **ARM**, IBM zSeries, Intel IA-64 …

Cpu Intel 80386 (anni ‘80)



- https://en.wikichip.org/wiki/intel/microarchitectures/80386
- 1.5 µm process
- 275,000 transistors
- Clock: 33 MHz

- **Esercizio: trovate gli stessi parametri per un processore moderno (> Intel i7)**

Linux OS - distribuzioni

- Per Linux OS si intende: kernel Linux più altro software (strumenti, librerie) che insieme formano un OS completo, pronto per l'uso => distribuzione (**distribution**)

- Linux kernel + progetto GNU = GNU/Linux

- Le prime distribuzioni apparvero nel 1992. La più vecchia ancora attiva è Slackware. Debian apparve nel 1993. Ubuntu (derivata da Debian) apparve nel 2004.

Standard linguaggio C

- Linguaggio C: lo standard “di fatto” era il libro “The C Programming language” di Kernighan e Ritchie (1978). C++ iniziò nel 1985.

- Necessità di standardizzare il linguaggio C: 1989, American National Standards Insitute (ANSI) C standard: definisce il linguaggio e la standard C library (funzioni stdio, stringhe, math, header files etc). Versione di C nota come **C89** (descritta nel libro “The C Programming language” , seconda edizione).
- Nel 1999 fu promulgato una versione aggiornata dello standard del linguaggio C: **C99**
- Nel 2011: **C11**
- Nel 2018: **C18**

POSIX

- POSIX: **Portable Operating System Interface**
- Gruppo di standard sviluppati da IEEE, per promuovere la portabilità di applicazioni a livello di codice sorgente.

- POSIX.1 : divenne uno standard di IEEE nel 1988; basato sulle UNIX system calls e API della C standard library. Ma non richiede necessariamente UNIX come OS (quindi altri SO possono implementare questo standard).

- POSIX.2: standardizza la shell ed altre utilities, inclusa l'interfaccia a riga di comando del compilatore C.

POSIX e Linux

- Nel 2001: POSIX.1-2001, noto anche come Single UNIX Specification Version 3: **SUSv3**. Sono **3700 pagine** di standard.

- 2008: **SUSv4** o POSIX.1-2008

- **Linux OS punta a seguire gli standard UNIX, ma non lo segue completamente**

Numeri di Linux Kernel

- Linux 5.4 sources:
- 66.031 files (git ls-files | wc -l)
- 27.679.764 lines (git ls-files | xargs cat | wc -l)
- 889.221.135 bytes (git ls-files | xargs cat | wc -c)

Tratto da https://bootlin.com/doc/training/embedded-linux/embedded-linux-slides.pdf

As of kernel version 4.6 (in percentage of number of lines)

- drivers/: 57.0%
- arch/: 16.3%
- fs/: 5.5%
- sound/: 4.4%
- net/: 4.3%
- include/: 3.5%
- Documentation/: 2.8%
- tools/: 1.3%
- kernel/: 1.2%
- firmware/: 0.6%
- lib/: 0.5%
- mm/: 0.5%
- scripts/: 0.4%
- crypto/: 0.4%
- security/: 0.3%
- block/: 0.1%
- ...

USER vs KERNEL

- Kernel e Utente sono due termini che vengono spesso utilizzati nei sistemi operativi.
- La loro definizione è piuttosto semplice: **il kernel è la parte del sistema operativo in esecuzione con privilegi più alti mentre lo spazio utente di solito si riferisce a applicazioni in esecuzione con privilegi bassi.**

- utente privilegiato: un utente autorizzato (e quindi affidabile) a svolgere funzioni rilevanti per la sicurezza che gli utenti ordinari non sono autorizzati a svolgere
- il privilegio è definito come la delega dell'autorità per eseguire funzioni rilevanti per la sicurezza su un sistema informatico (esempi di privilegi in un SO?)

USER vs KERNEL

- Modalità utente e modalità kernel sono termini che possono fare riferimento in modo specifico alla **modalità di esecuzione del processore**.

- Il codice che viene eseguito in modalità kernel può controllare completamente la CPU mentre il codice che viene eseguito in modalità utente presenta alcune limitazioni.

USER vs KERNEL

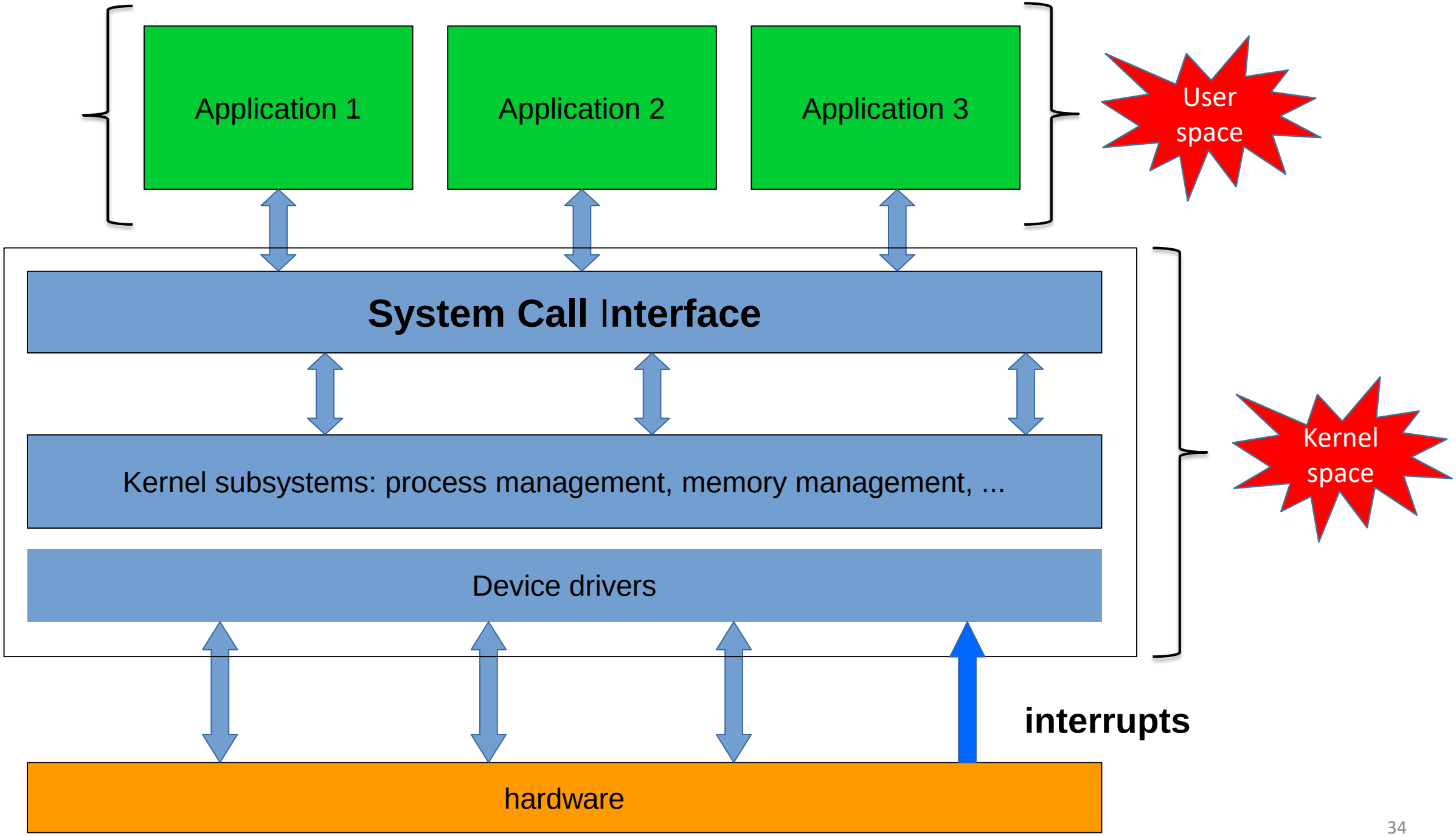
- Lo spazio utente e lo spazio del kernel possono riferirsi specificamente alla **protezione della memoria** o agli **spazi degli indirizzi virtuali** associati al kernel o alle applicazioni dell'utente.

- Semplificando grossolanamente, lo **spazio del kernel** è l'area di memoria riservata al kernel mentre lo **spazio dell'utente** è l'area di memoria riservata a un particolare processo dell'utente.

- Lo spazio del kernel è protetto dall’accesso diretto da parte delle applicazioni utente

Architettura tipica del sistema operativo

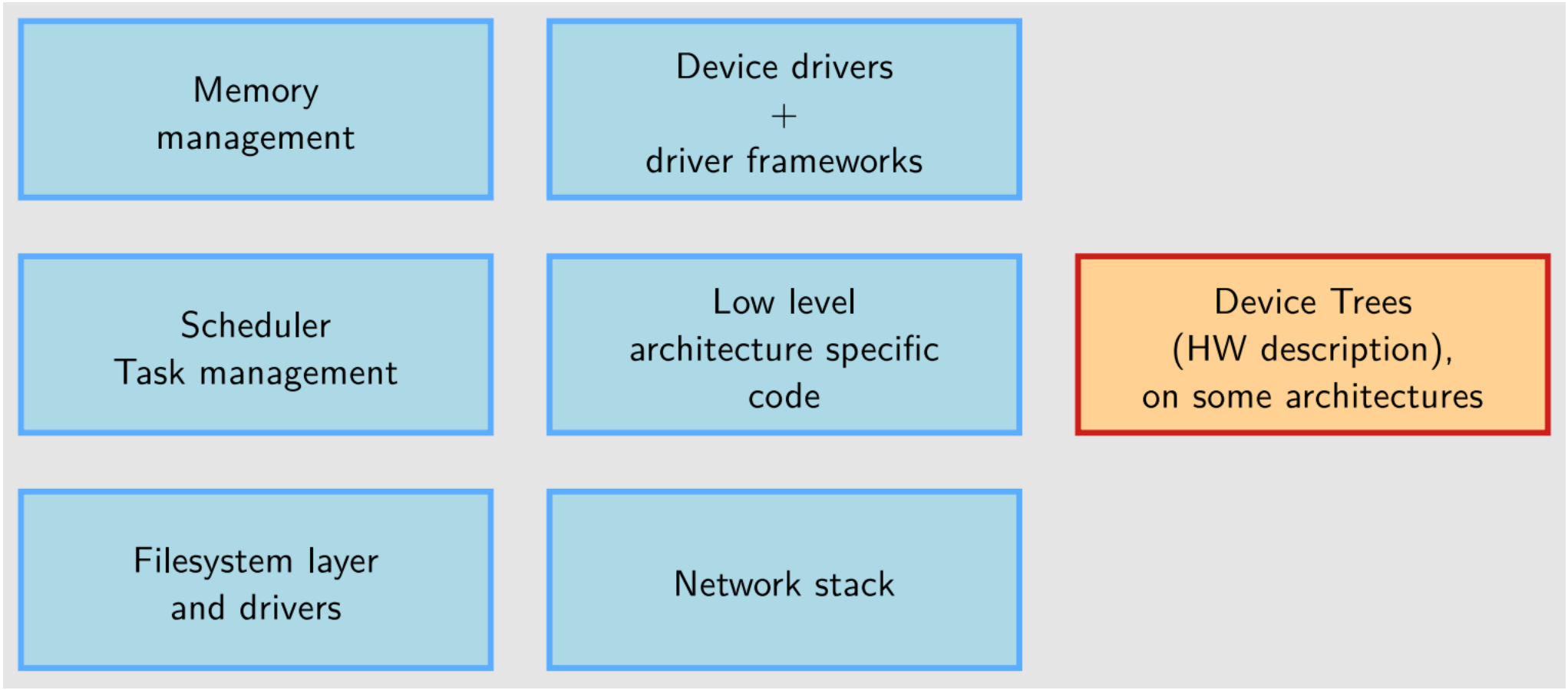
- Nell'architettura tipica del sistema operativo (vedere la figura seguente) il kernel del sistema operativo è responsabile dell'accesso e della condivisione dell'hardware in modo sicuro ed equo verso le applicazioni
- Il kernel offre alle applicazioni una Application Programming Interface (API) che generalmente viene chiamata «interfaccia delle chiamate di sistema» o «**system calls interface**»
- questa API è diversa dalle normali API di libreria perché rappresentano il limite in base al quale **la modalità di esecuzione passa dalla modalità utente alla modalità kernel**


Application 1
Application 2
Application 3
User space
System Call Interface
Kernel subsystems: process management, memory management, ...
Device drivers
Kernel space
interrupts
hardware

System call interface

- Per garantire la compatibilità delle applicazioni, le chiamate di sistema vengono modificate raramente. Linux lo impone in particolare (diversamente dalle API del kernel che possono cambiare in base alle esigenze)
- Il codice del kernel stesso può essere logicamente separato nel codice del kernel principale e nel codice dei driver di dispositivo. Il codice dei driver di dispositivo è responsabile dell'accesso a determinati dispositivi mentre il codice del kernel principale è generico.
- Il kernel principale può essere ulteriormente suddiviso in più sottosistemi logici (ad es. Accesso ai file, rete, gestione dei processi, ecc.)

Linux Kernel



https://bootlin.com/doc/training/embedded-linux/embedded-linux-slides.pdf