

Storia dei Sistemi Operativi

*potete avere a disposizione il computer più veloce dell'universo,
e il linguaggio di programmazione più espressivo di tutti i tempi,
ma provate ad usarli senza un sistema operativo decente...*

power is nothing, without control

Prima dei Sistemi Operativi

- Abbiamo visto come già nell'Analytical Engine di Charles Babbage fosse presente una idea di “programma” abbastanza simile alla nozione che ne abbiamo ai giorni nostri: una sequenza di istruzioni predefinite impartite alla macchina.
- A tale proposito, ecco una curiosità. Nel paragrafo 1.2 del Tanenbaum: I Moderni Sistemi Operativi, 3° ed., troviamo scritto:

*“Babbage capì che avrebbe avuto bisogno di **software** per il suo motore analitico, così assunse, come prima **programmatrice** al mondo, **una giovane donna** Ada Lovelace...”*

Meglio non fidarsi ciecamente di quello che troviamo scritto nei libri, anche se gli autori sono autorità del campo...

Prima dei Sistemi Operativi

- In effetti però, nei primi computer degli anni ‘40, il concetto di “programma” era assai lontano da quello moderno, e il concetto di “sistema operativo” era completamente sconosciuto.
- Ad esempio, “Programmare” l’ENIAC significava configurare a mano switch e collegamenti degli accumulatori e delle function tables.
- È con il modello di architettura von Neumann, a partire dall’EDVAC e dal Manchester Small Scale Experimental Machine che (ri)emerge finalmente in modo chiaro il concetto di **istruzione macchina**, da cui segue quello di programma come sequenza di istruzioni macchina.

Prima dei Sistemi Operativi

- Ma caricare un programma nella memoria di questi computer primitivi rimaneva ancora una attività sostanzialmente manuale:
- Nella Manchester Small Scale Experimental Machine le 32 word da 32 bit ciascuna venivano inizializzate con le istruzioni del programma da eseguire e i relativi dati di input usando 32 switch per settare a 0 o a 1 ogni singolo bit di ciascuna word.
- Nell'EDVAC il programma veniva letto da un nastro perforato ad hoc (più tardi da un nastro magnetico) e i dati potevano essere inseriti da una console a leve e pulsanti.
- **E l'idea di farsi aiutare da un programma per inserirne un altro nel computer e farlo girare era completamente aliena...**

L'era dei Sistemi Operativi

- A partire dagli anni ‘50 però le cose cominciano a cambiare rapidamente, sicché possiamo individuare otto fondamentali fasi nella storia dell’evoluzione dei Sistemi Operativi (SO):
 1. Job by Job / Open Shop pre SO era: ≈ fino al 1955
 2. Sistemi Batch seconda metà anni ‘50
 3. Multiprogrammazione inizio anni ‘60
 4. Timesharing metà anni ‘60
 5. Sistemi concorrenti fine anni ‘60
 6. Sistemi per Personal Computer seconda metà anni ‘70
 7. Sistemi Distribuiti anni ’80
 8. Sistemi per dispositivi mobili XXI secolo

L'era dei Sistemi Operativi

- Col tempo i sistemi operativi sono diventati sempre più sofisticati e complessi, in modo da soddisfare sempre meglio le diverse esigenze degli utilizzatori dei computer.
- Così, ai giorni nostri praticamente chiunque può usare un personal computer (ma anche un tablet o uno smartphone), senza per questo dover essere un esperto informatico.
- È il sistema operativo che trasforma l'oggetto più complesso mai progettato dall'uomo in uno strumento relativamente facile sa usare.
- Allo stesso tempo, sistemisti e programmatore sanno che il sistema operativo offre loro un ambiente di lavoro sofisticato, nel quale poter sfruttare al meglio le risorse del computer su cui stanno lavorando.

L'era dei Sistemi Operativi

- E c'è voluto del tempo per ideare e realizzare al meglio le tecniche che costituiscono l'essenza di un sistema operativo moderno: *time sharing*, *memoria paginata e virtuale*, *codice rilocabile*, e tante altre.
- Ma è stato anche necessario che i progettisti hardware avessero il tempo di incorporare nelle nuove CPU le caratteristiche necessarie ad implementare i servizi offerti dal sistema operativo (esempi?)
- Ed è stato necessario che la potenza di calcolo delle CPU crescesse in modo tale da compensare l'overhead richiesto dal SO per fornire i suoi servizi (esempi?)
- Perché il SO consuma tempo di CPU, che dunque non può essere impiegato per eseguire i nostri programmi. Ovviamente si ritiene⁷ che ne valga la pena...

L'era dei Sistemi Operativi

- Come per i linguaggi di programmazione, anche i Sistemi Operativi sono stati soggetti ad un processo di selezione naturale, dal quale pochi sono sopravvissuti.
- Essenzialmente, Windows e Unix con le loro tante varianti, a cui negli ultimi anni si sono aggiunti i sistemi operativi per dispositivi mobili, come iOS, e Android. Ma questi ultimi – come del resto anche Windows e Unix – derivano in larga parte da sistemi operativi sviluppati in precedenza.
- La tecnologia dei sistemi operativi, il corpo di conoscenze che permette di sviluppare un buon sistema operativo, si è stabilizzato tra gli anni ‘70 ‘80, e i nuovi sistemi operativi non hanno fatto altro che inglobare il risultato di esperienze precedenti. A volte, naturalmente, migliorando il risultato finale.

Fase 1: Job by Job / Open Shop

- Fino ai primi anni '50 l'idea di Sistema Operativo non esisteva nemmeno, e i computer erano usati fondamentalmente solo da coloro che li avevano costruiti e dai loro collaboratori.
- Dunque, far girare un programma sui computer primitivi di quei tempi consisteva più o meno nei seguenti passi:
 1. Scrivere il programma su carta in codice macchina binario, decimale, ottale o esadecimale. Il codice usava indirizzi assoluti, e il concetto di codice rilocabile era ancora di là da venire.
 2. Codificare il programma su schede perforate (o su nastro), mediante opportuni perforatori elettromeccanici (ma ricordate che prima ancora il programma doveva essere inserito a mano in memoria configurando bit per bit opportuni switch).

Fase 1: Job by Job / Open Shop

3. Nello slot prenotato di 15/30 minuti il programmatore si recava nella sala del computer sperando che il computer non fosse guasto (evento molto frequente in quegli anni).
4. Una buona parte dello slot di tempo assegnata al programmatore serviva per preparare adeguatamente il computer, in particolare predisporre le schede perforate nel lettore di schede che trasferiva il programma nella memoria del computer.

In alternativa, il contenuto delle schede veniva riversato sul nastro magnetico che faceva da dispositivo di input del sistema.

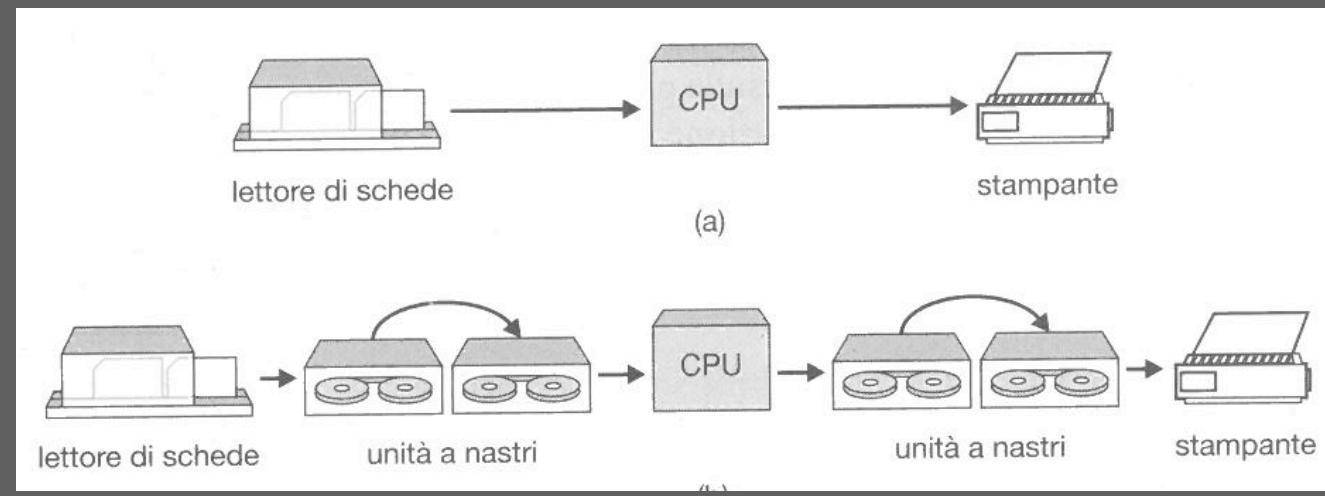
5. Finalmente il programmatore poteva sedersi alla console del computer e comandare l'esecuzione del programma



OPERATING THE UNIVAC 1. REMINGTON RAND USA 1952

Fase 1: Job by Job / Open Shop

6. il contenuto di ogni singola cella di memoria poteva essere letto specificandone l'indirizzo con opportuni switch. In questo modo era possibile seguire l'andamento del programma, e leggere i risultati dell'esecuzione di un calcolo.
7. L'output del programma poteva essere stampato su una telescrivente, oppure riversato su nastro magnetico e da qui stampato o convertito di nuovo in schede perforate.



Fase 1: Job by Job / Open Shop

- Dunque, molto tempo veniva speso per la gestione manuale di schede perforate e nastri magnetici. In più, se qualcosa andava storto e non veniva prodotto nessun output il programmatore era costretto ad analizzare il contenuto della memoria per capire cosa era successo.
- A metà anni ‘50, lo sviluppo dei primi assembler e dei primi linguaggi ad alto livello rese le cose ancora più laboriose. Per far girare un programma scritto ad esempio in assembler occorreva:

Caricare da nastro l’assemblatore e il programma da assemblare.

Salvare il risultato della compilazione su un nastro di output (o su una serie di schede perforate)

Usare il nastro di output (o il gruppo di schede perforate) per caricare in memoria l’eseguibile e farlo finalmente girare.

Fase 1: Job by Job / Open Shop

- La necessità di gestire in modo per quanto possibile efficiente le schede perforate e i nastri magnetici portò all'introduzione di una figura professionale addetta alla sala macchine.
- Ora, il programmatore affidava il pacchetto di schede col proprio programma all'addetto, il quale avrebbe eseguito tutti i passi necessari per eseguire il programma, stampare l'output e consegnarlo al programmatore.
- Se il programma terminava in modo anomalo, l'addetto poteva far stampare il contenuto della memoria del computer e consegnarlo al programmatore il quale, con un po' di pazienza, poteva così risalire alle cause del malfunzionamento.

Fase 1: Job by Job / Open Shop

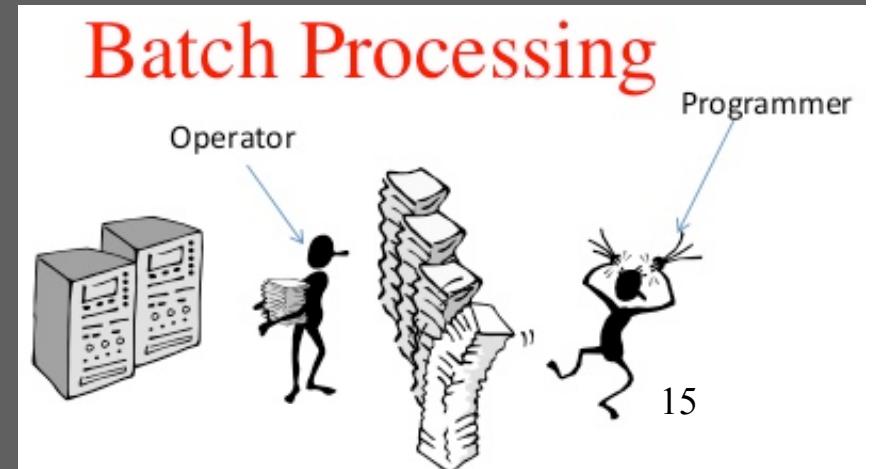
- Notate dunque che in questa fase non c'era sostanzialmente nessuna interfaccia tra la macchina e i programmi degli utenti:
- Ogni indirizzo in memoria, e dunque ogni cella di memoria erano raggiungibili (cioè potevano essere usati in un programma)
- Non esisteva alcun *File System*, e nemmeno il concetto di *file*: le informazioni (programmi più dati di input) venivano caricati direttamente da schede perforate o da nastro magnetico. L'output di un programma finiva direttamente su schede o nastro.
- nella memoria del computer veniva caricato e girava un solo programma alla volta.

Fase 2: Sistemi Batch

- Ma il modo di operare dell'addetto suggerì un'idea che, per quanto oggi ci sembri banale, probabilmente è la più importante innovazione nella storia dei sistemi operativi:

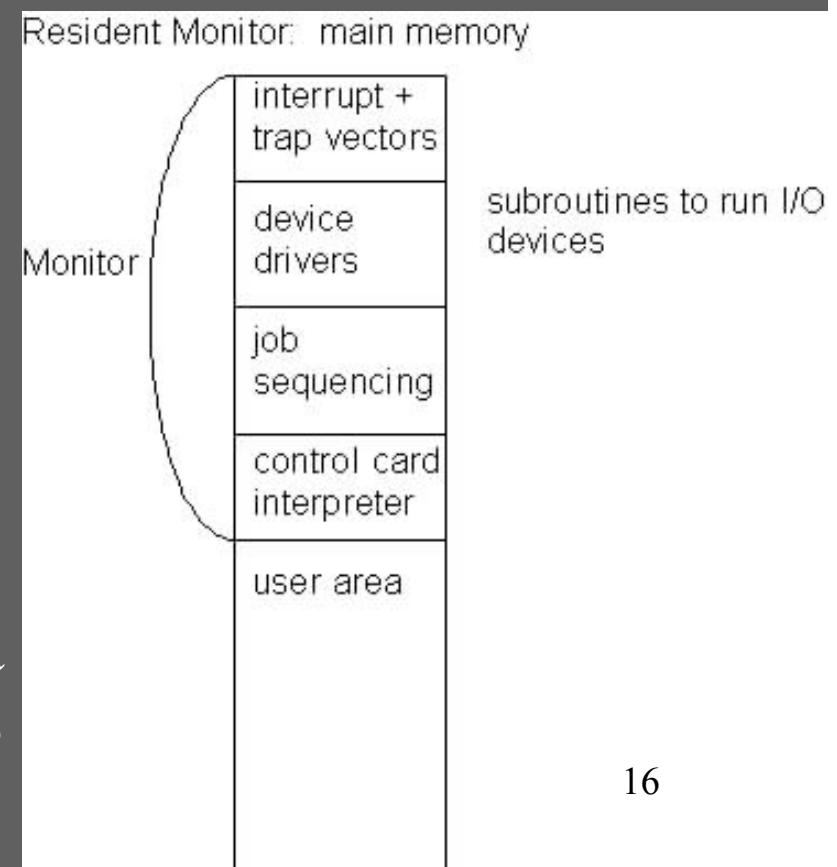
Permettere al computer di pianificare il lavoro da fare (ossia, eseguire i programmi) per mezzo di un programma apposito.

- Dunque, l'addetto poteva raccogliere i pacchetti di schede (a volte chiamati **job**) dei programmi da eseguire e inserirli tutti assieme nel lettore di schede, separati da opportune schede di controllo.
- Le schede di controllo avrebbero dovuto indicare l'inizio e la fine di ogni pacchetto/programma, e cosa fare di quel particolare pacchetto.



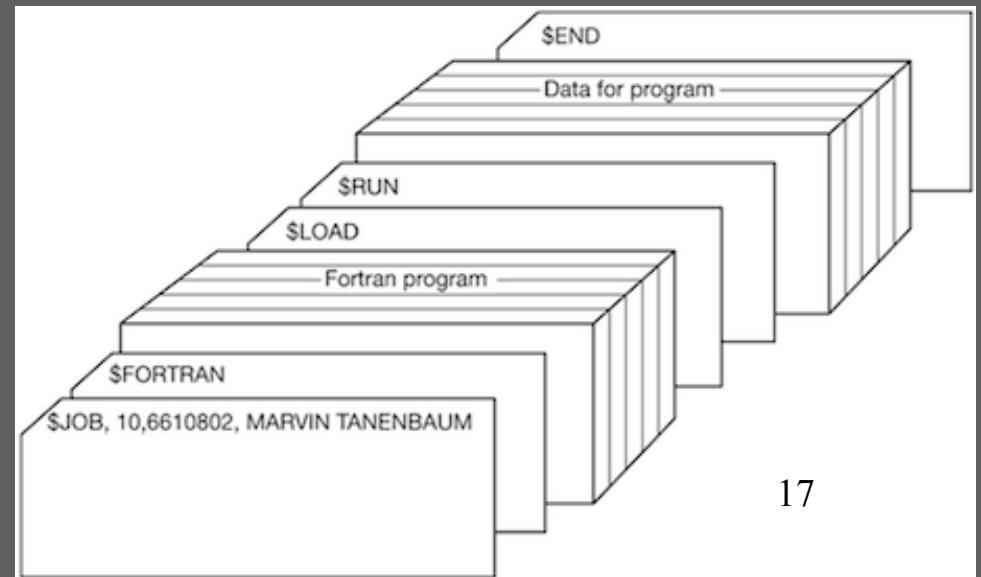
Fase 2: Sistemi Batch

- Le istruzioni specificate nelle schede di controllo sarebbero state eseguite da un programma speciale sempre residente in memoria, non a caso chiamato **Resident Monitor**: *nasceva dunque l'idea di un programma scritto per facilitare l'esecuzione di altri programmi.*
- Naturalmente il Resident Monitor doveva essere in grado di:
 - Capire i comandi impartiti dalle schede di controllo
 - Gestire i dispositivi di input/output
 - Far partire un programma e alla sua terminazione riprendere il controllo della macchina.



Fase 2: Sistemi Batch

- L'uso congiunto di schede perforate e di uno o più dispositivi a nastro magnetico rendeva il lavoro del Resident Monitor ancora più efficiente. Ecco il layout di un gruppo di schede per far girare un programma scritto in Fortran (notate le schede con angolo tagliato):
- **\$ FORTRAN**: Carica dal nastro di sistema il compilatore Fortran e compila il programma sorgente. L'eseguibile era di solito salvato su un nastro di supporto (in inglese, uno *scratch tape*).
- **\$LOAD**: Carica in memoria il programma utente (di default, dallo scratch tape)
- **\$RUN**: esegui il programma sui dati fino alla scheda **\$END**

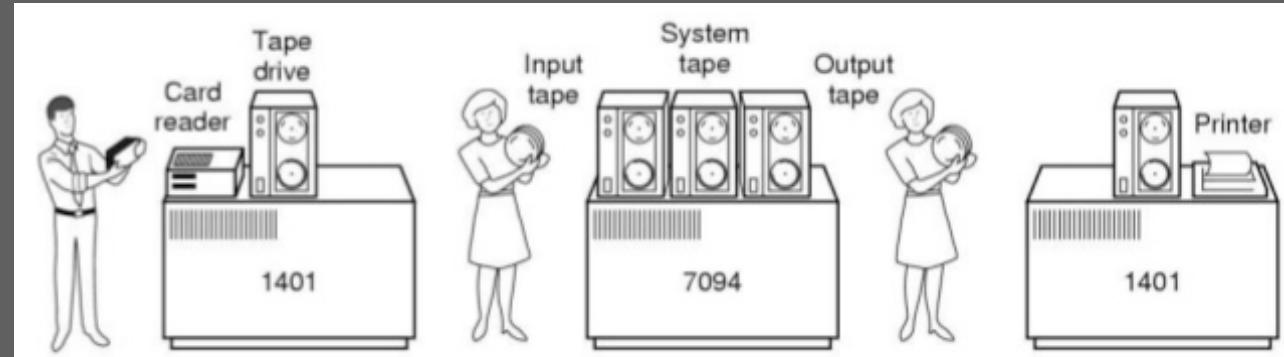


Fase 2: Sistemi Batch

- Rispetto alla Fase 1 qualcosa era cambiato: ora due programmi risiedevano contemporaneamente in memoria, e cominciava ad emergere una distinzione tra un programma e i suoi dati.
- Tuttavia, nessuna protezione della memoria era disponibile, e se un programma girava male era sempre molto difficile capire cosa era successo, e spesso era necessario dover resettare l'intero sistema.
- Senza l'intervento specifico di un operatore comunque, *il Resident Monitor permetteva di automatizzare l'esecuzione in sequenza di un gruppo di programmi*, nel loro complesso indicati come **batch jobs**.
- Per estensione, si indicò quindi con **batch processing** il lavoro del Resident Monitor per eseguire tutti i programmi di un certo *batch*, e **batch system** erano quei computer su cui girava il Resident Monitor.

Fase 2: Sistemi Batch

- Più programmi da eseguire potevano essere raccolti su nastro magnetico, e più si mitigava la lentezza dei dispositivi di input e output, sovrapponendo l'I/O (su un computer “satellite”) con l'esecuzione dei programmi in sequenza (sul computer principale).
- Dunque, il modo di procedere era più o meno il seguente: i pacchetti di schede dei vari programmi venivano raccolti e inseriti nel lettore di schede, e i job venivano trasferiti su nastro magnetico.
- Il nastro magnetico veniva trasferito a mano sul lettore di nastro collegato al computer e i job eseguiti in sequenza. Nel frattempo, altri job potevano essere riversati dal lettore di schede ad un nuovo nastro.



Fase 2: Sistemi Batch

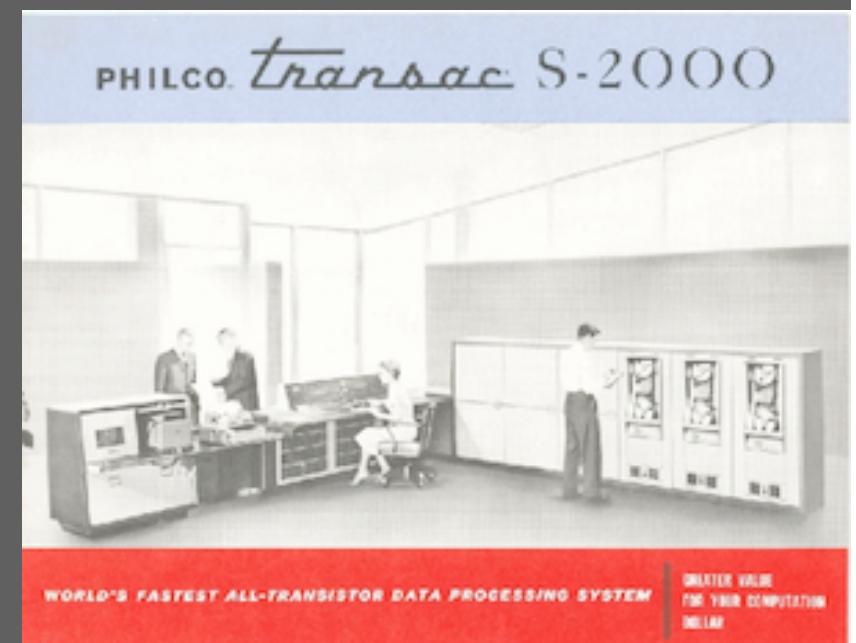
- Analogamente, l'output di ogni job veniva scritto su un nastro magnetico, il quale veniva poi trasferito su un lettore di nastro separato e collegato alla stampante, in modo da poter stampare in output i risultati dell'esecuzione dei vari job.
- La produttività del sistema aumentava, ma a scapito dei tempi di attesa del singolo utente: era l'addetto alla sala macchine a riorganizzare i job e a decidere in che ordine eseguirli...
- Dunque, un programmatore poteva dover aspettare ore, o anche giorni, prima di poter entrare in possesso della stampa dell'output del proprio programma, output che poteva anche ridursi ad un messaggio di errore del compilatore dovuto ad un punto e virgola messo male!

Fase 2: il sistema BKS

- Un tipico esempio di sistema Batch è il **BKS**, sviluppato per il **Philco Transac 2000**, un computer messo in commercio nel 1957, e uno dei primi ad essere costruito completamente a transistor.
- (Nota a margine: Nel 1959 la **Olivetti** mette in commercio il suo **Elea 9003: ELaboratore Elettronico Automatico**. Costruito solo con transistor, precedette di molti mesi la commercializzazione del primo computer IBM tutto a transistor, l'IBM 7090).
- Il Philco 2000 era dotato di 32768 word di memoria, di cui 2688 occupate dal BKS. Il sistema era in grado di prendere l'input da nastro magnetico, o anche direttamente dal lettore di schede perforate (nel caso il lettore di nastro non fosse disponibile)

Fase 2: il sistema BKS

- In realtà, il Philco 2000 era dotato di più lettori di nastro, che sostanzialmente funzionavano come aree diverse di un file system: un nastro per i *dati temporanei* dei programmi, uno per le *librerie*, uno per i *dati di output*, uno per i *programmi da eseguire*, e così via.
- I processori di quegli anni non fornivano alcun supporto hardware per la protezione della macchina da usi impropri, e quindi ai programmatori si raccomandava di scrivere programmi che:
 - non contenessero istruzioni di halt
 - non usassero istruzioni per riavvolgere i nastri
 - non indirizzassero locazioni di memoria riservate al sistema



Fase 3: multiprogrammazione

- L'efficienza dei sistemi batch era fortemente limitata dall'uso di supporti di memorizzazione di massa lenti e ad accesso esclusivamente sequenziale, schede perforate e nastri magnetici (a cui spesso si aggiungeva l'intervento umano, ancora più lento).
- Inoltre, come abbiamo visto nell'esempio del BKS, l'assenza di meccanismi hardware di protezione non permetteva al resident monitor di proteggere la macchina da usi impropri.
- A partire da fine anni '50 / inizio degli anni '60 però, diverse innovazioni tecnologiche diedero il via ad una nuova fase nella storia dei sistemi operativi.

Fase 3: multiprogrammazione

- In particolare, furono di enorme importanza l'introduzione di **dispositivi di memorizzazione di massa ad accesso diretto**: le memorie a tamburo rotante prima e successivamente gli hard disk.
- E per l'attività del sistema operativo, fu fondamentale la realizzazione di canali dati diretti tra memoria primaria e secondaria e la diffusione di CPU a transistor più potenti e in grado di implementare un sofisticato sistema di interrupt.
- Tutto ciò permetteva di simulare l'esecuzione “contemporanea” di più programmi e più operazioni di Input e Output, ciò che appunto chiamiamo la **multiprogrammazione**.

Fase 3: multiprogrammazione

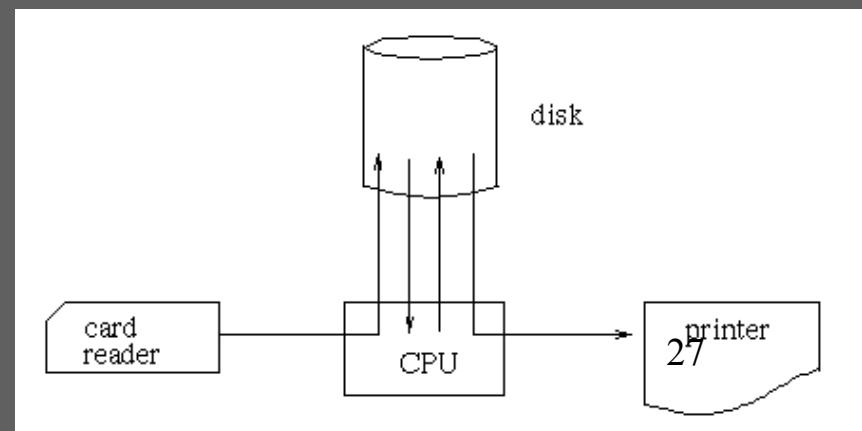
- Il termine multiprogrammazione compare per la prima volta in un fondamentale articolo del 1959 di **Christopher Strachey**, il quale osservò in seguito che l'articolo trattava principalmente di come evitare le attese dovute alle periferiche.
- Infatti usando nastri magnetici, mentre il contenuto di un pacco di schede perforate veniva trasferito all'estremità di un nastro, la CPU non poteva ovviamente leggere “contemporaneamente” ciò che era memorizzato all'altra estremità del nastro (in realtà avrebbe potuto, ma con tempi di attesa inaccettabili).
- Ma con un hard disk ciò è possibile: la testina di lettura/scrittura si può spostare avanti e indietro fra aree diverse del disco, portando avanti “in parallelo” diverse operazioni di lettura/scrittura. 25

Fase 3: multiprogrammazione

- L'hard disk dunque poteva essere usato come un buffer molto capiente: un'area di memoria nella quale depositare il contenuto di un pacchetto di schede perforate man mano che queste venivano lette e l'output di un programma man mano che questo veniva prodotto.
- Quando un pacchetto di schede fosse stato completamente trasferito sull'hard disk, il programma relativo poteva partire. Quando un programma che stava girando avesse completato la sua esecuzione, l'output memorizzato sull'hard disk avrebbe potuto essere stampato.
- E grazie all'hard disk, la cui testina di lettura si può spostare velocemente da un punto all'altro del disco magnetico (e grazie al sistema di gestione degli interrupt) le due operazioni potevano avvenire “contemporaneamente”: non era più necessario completarne una prima di avviare l'altra.

Fase 3: multiprogrammazione

- Questa modalità asincrona di eseguire i programmi e di gestire le operazioni di input e output veniva chiamata **spooling**, dalle iniziali di: **s**imultaneous **p**eripheral **o**peration **on** **l**ine.
- Da notare che non dovendo più eseguire i programmi nell'ordine in cui questi erano stati memorizzati su nastro magnetico, diveniva possibile deciderne l'ordine di esecuzione in base a criteri di efficienza, ad esempio scegliendo prima i più corti.
- Nota: oggigiorno a volte il termine *batch processing* viene usato come sinonimo di *spooling*, ma come abbiamo appena visto, questo è improprio.



Fase 3: multiprogrammazione

- Dunque, il termine “multiprogrammazione” fu coniato per intendere qualcosa di parzialmente diverso da come lo intendiamo oggi:, indicava appunto la possibilità di eseguire in modo asincrono (e dunque anche di sovrapporre) diverse operazioni di input e output.
- Ma in effetti, fu poi naturale generalizzare il concetto, di modo che se il programma in esecuzione si doveva fermare temporaneamente per eseguire delle operazioni di I/O, un altro programma poteva essere mandato in esecuzione.
- Questo modello di esecuzione dei programmi è noto come multitasking, o più propriamente **multitasking cooperativo**, in quanto un programma può girare grazie al fatto che un altro programma lascia temporaneamente la CPU in modo volontario⁸.

Fase 3: il sistema Atlas

- Il supervisore **Atlas** fu presentato nel 1961, e costituiva il sistema operativo dell'**Atlas Computer**, progettato all'Università di Manchester (GB) in collaborazione con la Ferranti Computers.
- Sviluppato da **Tom Kilburn**, **Bruce Payne** e **David Howard**, l'Atlas è universalmente riconosciuto come il primo sistema operativo moderno, e rappresentò probabilmente il più importante passo avanti nella storia dei sistemi operativi.
- Nell'Atlas compaiono due fondamentali concetti dei sistemi moderni: le **system call** (originariamente chiamate **extracode**): codice del sistema operativo che i programmi usano per eseguire operazioni “delicate” (ad esempio le operazioni di I/O) e l'implementazione della **memoria virtuale con paginazione su richiesta**.



Fase 3: il sistema Atlas

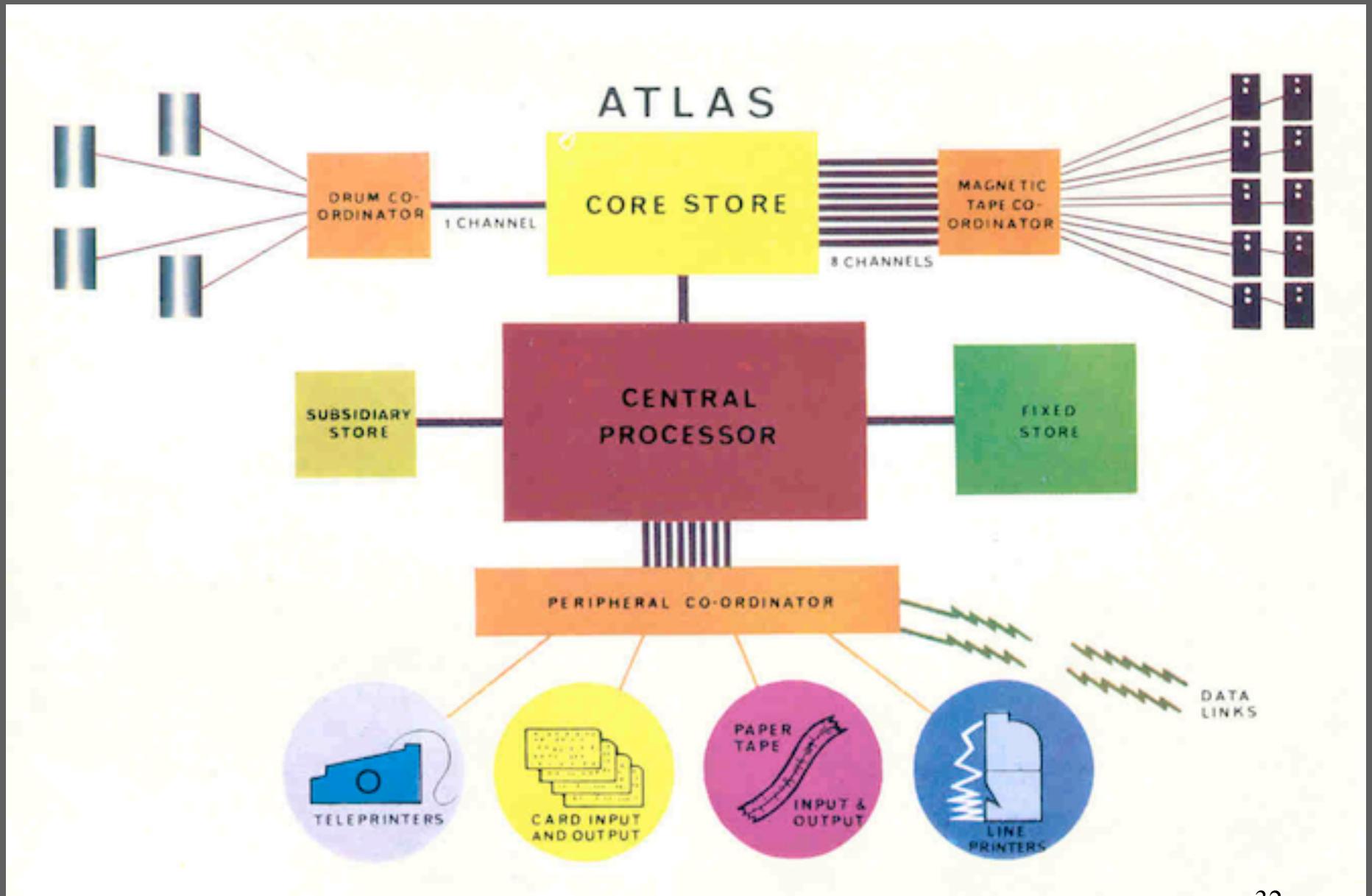
- Nell'Atlas si fa strada l'idea che il sistema operativo venga invocato di frequente, e dunque si alterni spesso con l'esecuzione del codice dei programmi utente. Dall'articolo originale sull'Atlas, del 1961:

“the supervisor therefore becomes active on frequent occasions and for a variety of reasons – in fact whenever any part of the system requires attention from it. [...] The central computer thus shares its time between these supervisor activities and the execution of object programs is such that there is mutual protection between object programs and all parts of the supervisor. The supervisor program consists of many branches which are normally dormant but which can be activated whenever required. The sequence in which the branches are activated is essentially random, being dictated by the course of an object program and the functioning of the peripheral equipments.”
- In sostanza, l'essenza di un sistema operativo in poche frasi... ³⁰

Fase 3: il sistema Atlas

- La memoria primaria dell'Atlas era suddivisa in pagine (chiamate **blocks**) da 512 words a 48 bit, ed era così composta:
- **fixed store**: una memoria a nucleo magnetico (MCM) da 8.192 words per il codice del supervisor
- **subsidiary store**: una MCM da 1024 words per i dati del supervisor
- **core store**: una MCM da 16.384 words per il processo utente in esecuzione
- **drum store**: memoria a tamburo rotante da 98.304 words per i processi non in esecuzione (ad esempio perché in attesa del completamento di una operazione di I/O).

Fase 3: il sistema Atlas



Fase 3: il sistema Atlas

- Per poter essere eseguito, un programma veniva letto da uno dei nastri magnetici di input e trasferito nella core store.
- Data la capacità limitata della core store, codice e dati erano copiati anche nel **system input tape**, usato dal supervisor come **area di swap** durante l'esecuzione del programma.
- Quando il programma in esecuzione si fermava per una operazione di I/O le sue pagine venivano trasferite dalla core store alla drum store, e al suo posto veniva fatto partire un altro programma.
- A regime, e in attesa di poter tornare in esecuzione nella core store, più programmi si potevano trovare nella drum store. Se questo si riempiva alcune pagine potevano essere salvate nel system input tape dal supervisor, e poi sovrascritte.
33

Fase 3: il sistema Atlas

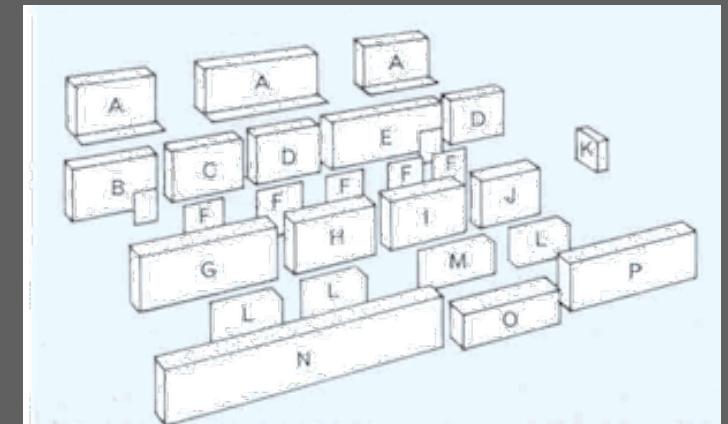
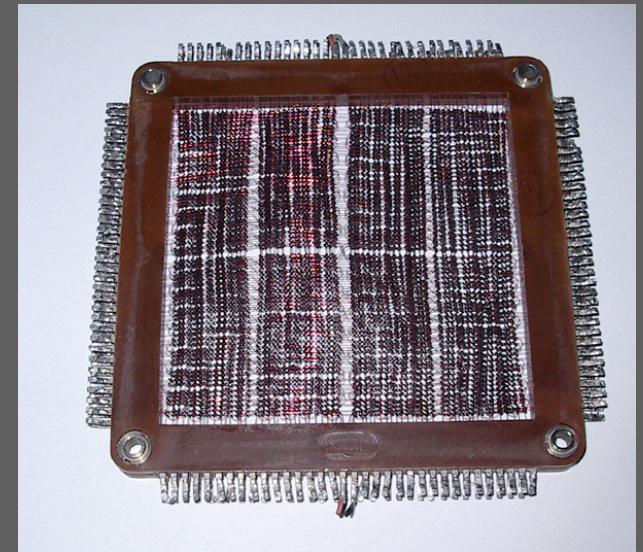
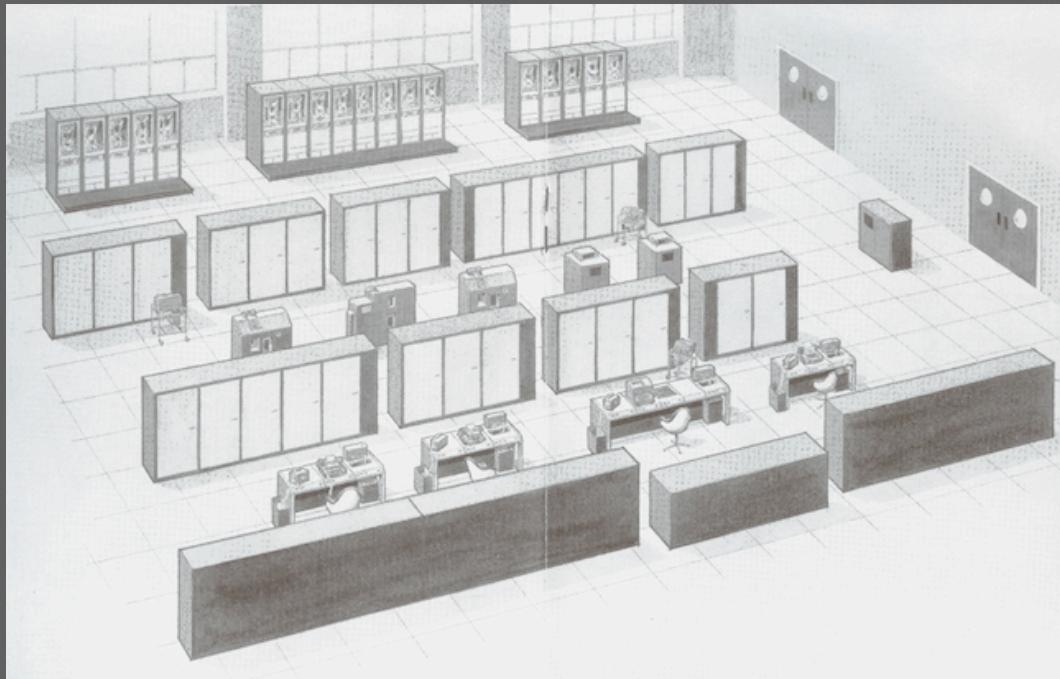
- Se il programma in esecuzione indirizzava una pagina non nella core store ma presente nella drum store, questa veniva trasferita nella core store dal supervisor, ma senza alcun context switch tra processi utente.
- Se invece la pagina mancante era presente solo nel system input tape allora il programma che aveva generato il page fault veniva spostato nella drum store in attesa del prelievo dal nastro della pagina richiesta. Nel frattempo veniva eseguito un altro programma.
- Il supervisor si assicurava di avere sempre un frame libero nella core store, eventualmente scegliendo una pagina vittima mediante un algoritmo LRU e salvandola se necessario nella drum store

Fase 3: il sistema Atlas

- Dunque, nell'Atlas vediamo emergere una sofisticata gestione della memoria, organizzata in una gerarchia che è valida ancora oggi, sebbene implementata con tecnologie molto più efficienti.
- La core store dell'Atlas, piccola e veloce, veniva usata sostanzialmente come una **memoria cache** contenente una porzione del programma in esecuzione (o anche tutto, se ci stava).
- La drum store svolgeva il ruolo di vera e propria memoria principale, ospitando i programmi in attesa di poter usare la CPU
- Uno dei nastri a disposizione del sistema (l'Atlas era in grado di gestirne fino a un massimo di 32) veniva riservato al supervisor, e usato come area di swap.

Nota a margine: le dimensioni dell'Atlas

- A destra, un banco di memoria a nucleo magnetico usato nell'Atlas: misurava 12×12 cm e memorizzava 64×64 bit di dati. Sotto: una tipica installazione Atlas.



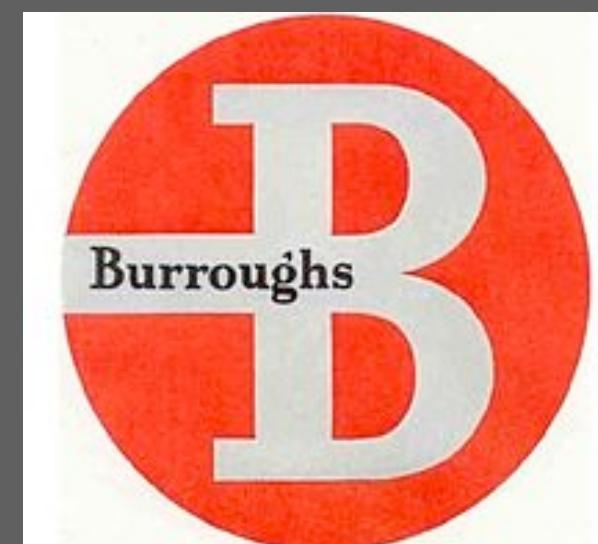
- D: core store; E: core store coordinator; I: central computer; J: fixed store; K: core store power supply; N: general power supply.

Fase 3: il sistema B5000

- Sempre nel 1961, l'americana **Burroughs Corp.** presenta il **B5000**, il primo computer in grado di gestire, mediante opportune istruzioni e registri dedicati, uno **stack**, una innovazione architetturale che verrà adottata da tutte le future architetture.
- In sostanza, ad ogni processo veniva associata un'area di memoria principale privata gestita a stack dove salvare valori temporanei e passare parametri e risultati delle subroutine, il che rendeva l'esecuzione dei programmi particolarmente efficiente.
- Il **Master Control Program (MCP)** del B5000 fu il primo sistema commerciale ad implementare una memoria virtuale segmentata, e il primo a soffrire (in media una volta al giorno) di **thrashing**, che veniva risolto semplicemente facendo ripartire la macchina. ³⁷

Fase 3: il sistema B5000

- Ma soprattutto, l'MCP va ricordato perché fu il primo sistema operativo ad essere completamente scritto in un **linguaggio ad alto livello**, una variante dell'Algol, una scelta davvero rivoluzionaria per quei tempi, che mostrava però che la cosa era possibile.
- Nell'MCP compare anche l'uso di un **timer hardware** usato dal sistema operativo per proteggersi da programmi utente che non rilasciavano mai la CPU: allo scadere del timer un interrupt restituiva il controllo all'MCP che terminava il programma in questione.
- Il B5000 poteva anche essere configurato con due processori, il che fa dell'MCP anche il primo sistema operativo multi-processore.



Fase 4: timesharing

- È piuttosto interessante che l'idea del timesharing nasca più o meno contemporaneamente a quella della multiprogrammazione. In un report datato 1 gennaio 1959 John McCarthy scrive:

“I want to propose an operating system [for the IBM 709] that will substantially reduce the time required to get a problem solved on [...] time-sharing. That is, the computer must attend to other customers while one customer is reacting to some output”
- McCharty qui si riferiva a ciò che oggi noi chiamiamo programmi interattivi, che appunto interagiscono con l'utente attraverso una alternanza di operazioni di input (ad esempio scrivendo sulla tastiera) e di output (ad esempio leggendo qualcosa sullo schermo).
- Nel 1961 McCharty chiarisce ancora meglio la sua visione:

Fase 4: timesharing

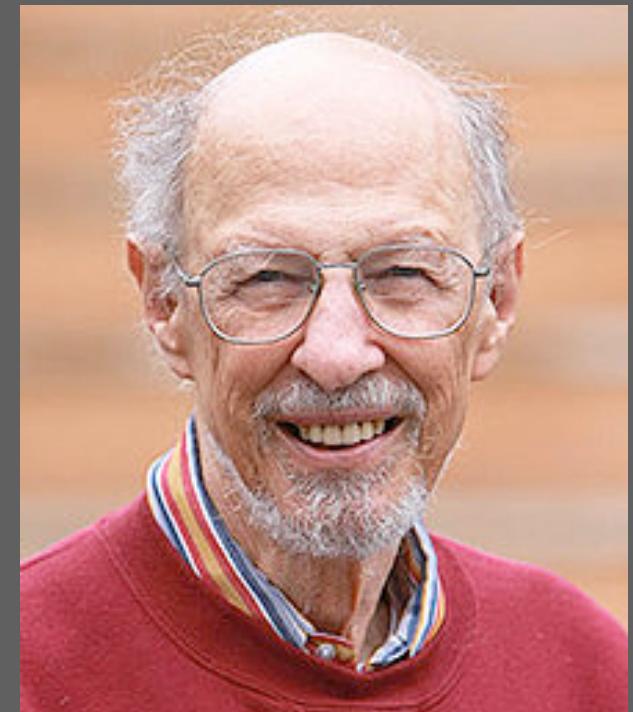
“By a time-sharing computer system I shall mean one that interacts with many simultaneous users through a number of remote consoles. Such a system will look to each user like a large private computer [...] when the user wants a service he simply starts typing in a message requesting the service. The computer is always ready to pay attention to any key that he may strike. [...]”

Because programs may do only relatively short pieces of work between human interactions, it is uneconomical to have to shuttle them back and forth continually to and from secondary storage. [...] The final requirement is for secondary storage large enough to maintain the users’ files so that users need not have separate card or tape input-output units.”

- Tutto ciò oggi a noi sembra ovvio, ma nel 1961 nessuno aveva mai ancora neanche immaginato una cosa simile...

Fase 4: il sistema CTSS

- Il primo sistema operativo timesharing è più o meno coevo del primo sistema multitasking: si tratta del **Compatible Time-Sharing System**, meglio noto come **CTSS**, sviluppato nel 1961 da **Fernando Corbatò**, **Marjorie Merwin-Dagget** e **Robert Daley** al **MIT** (Massachusetts Institute of Technology) di Cambridge.
- L'obiettivo esplicito del progetto era quello di permettere a più persone di usare il computer (l'IBM 709 prima, inizialmente, il 7090 poi) per mezzo di telescriventi, attraverso un supervisor che alternava l'esecuzione dei vari programmi, un breve burst di CPU per ciascuno.
- (In figura, Fernando José Corbatò, premio Turing nel 1990 per le sue ricerche sui sistemi operativi time sharing)



Fase 4: il sistema CTSS

- Il sistema CTSS cercava di garantire che ogni utente ricevesse risposta dal computer in tempi ragionevoli. Questa da sola era una vera e propria novità, per quei tempi.
- Ma soprattutto, la possibilità di interagire col sistema, specialmente durante la fase di debugging di un programma, rappresentava un enorme passo avanti rispetto al modo di fare girare i programmi di quegli anni.
- I progettisti del CTSS si trovarono a dover risolvere problemi che, tutti insieme, non erano mai affrontati prima, e che oggi costituiscono l'essenza di un sistema operativo moderno. Eccone qui di seguito alcuni.

Fase 4: il sistema CTSS

1. Per garantire l'interattività i programmi in esecuzione dovevano essere tenuti tutti contemporaneamente in memoria primaria: il CTSS doveva tenere traccia delle diverse aree di memoria occupate dai programmi, e garantirne la protezione.
2. I programmi potevano dover essere spostati dalla memoria primaria alla secondaria e viceversa. Solo codice dinamicamente rilocabile garantiva che ciò potesse essere fatto con una certa efficienza.
3. Il programma in esecuzione doveva poter essere interrotto temporaneamente alla fine del suo quanto di tempo, e ovviamente riprendere dal punto in cui era stato fermato.

Fase 4: il sistema CTSS

4. Le operazioni di I/O dovevano essere eseguite sotto il controllo del sistema, per evitare usi impropri di risorse condivise.
5. In caso di operazioni di I/O lente (ad esempio quelle che usavano un lettore di nastro) il CTSS doveva essere pronto a spostare il programma utente in memoria secondaria, implementata essa stessa su nastri magnetici.
6. Gli utenti dovevano poter interrompere un loro programma utente malfunzionante dalla loro postazione di lavoro.
7. Gli utenti dovevano poter interrogare il sistema per sapere, ad esempio, quale programma stava girando in quel momento, o se un certo dispositivo di I/O era o no disponibile.

Fase 4: il sistema CTSS

- Nella versione iniziale il CTSS era in grado di gestire 3 utenti “on line”, più un quarto utente “batch” che richiedeva solo di far girare un programma in background. Nelle versioni successive il numero di utenti collegati contemporaneamente fu portato fino a 32.
- Dei circa 150 Kbyte di core memory del 7090, circa 120 Kbyte erano assegnati ai processi utente, e i restanti 30 Kbyte al supervisor. Il 7090 aveva un meccanismo di protezione della memoria, e ciò permetteva al CTSS di tenere in core memory più processi contemporaneamente.
- Dal punto di vista degli utenti, la cosa più rivoluzionaria era forse costituita dalla possibilità di dialogare col supervisor alla telescrivente mediante opportuni comandi. Questo filmato del 1964 ci dà un’idea di quanto tutto ciò fosse considerato innovativo:
www.youtube.com/watch?v=sjnmckVnLi0

Fase 4: il sistema CTSS

- Oltre al timesharing e all'interattività col sistema, Il CTSS introdusse anche altre idee che sono diventate scontate per tutti, ai giorni nostri.
- Il CTSS era dotato di **accesso controllato degli utenti** attraverso una fase di login e password. Questo era necessario dato che gli utenti non dovevano più recarsi alla sala del computer per usarlo.
- Gli utilizzatori del sistema incominciarono a sviluppare **nuovi comandi**, che arricchivano il CTSS aggiungendosi a quelli esistenti.
- Gli utenti del CTSS cominciarono ben presto a comunicare fra loro lasciando nel sistema file con nomi del tipo “TO_JOHN”. Nel 1965 fu scritto un programma “**MAIL**” per automatizzare lo scambio di *mail* tra gli utenti. Nel 1971 fu usato per inviare il primo SPAM... 46

Fase 4: il sistema MULTICS

- Ma il CTSS va ricordato anche perché fu la base a partire dalla quale prese vita il progetto dello sviluppo del **MULTICS** (**MULT**ipleplexed **I**nformation **and** **C**omputing **S), il sistema operativo più ambizioso e controverso degli anni '60.**
- Lo sviluppo del MULTICS inizia nel 1964 all'interno del **Project MAC**, un laboratorio di ricerca del MIT, diretto da **Robert Fano** (l'abbiamo visto nel filmato) e in cui lavoravano anche Fernando Corbatò, Marvin Minsky e John McCarthy.
- L'obiettivo era progettare un sistema operativo capace di soddisfare praticamente tutte le necessità presenti (ma anche degli anni a venire) degli utenti di un sistema di calcolo di grandi dimensioni. (In figura, Robert Fano)



Fase 4: il sistema MULTICS

- Al progetto partecipavano anche la **General Electric** e i **Bell Labs**. Questi ultimi però si tirarono fuori dal progetto nel 1969, in occasione del completamento del MULTICS stesso.
- Anche la General Electric abbandonò l'impresa, ma l'anno successivo fu acquisita dalla Honeywell che si decise a commercializzare il MULTICS, che infatti fu installato da qualche decina di università e compagnie nei loro centri di calcolo. L'ultimo MULTICS è stato disattivato nell'anno 2000, in Canada.
- Sorprendentemente però, esiste ancora una piccola comunità on-line di **multicians**, appassionati che discutono del sistema e “ricordano i bei tempi”: <http://www.multicians.org>

Fase 4: il sistema MULTICS

- In effetti il MULTICS non ebbe un gran successo commerciale, probabilmente proprio perché era decisamente un sistema esagerato. Ecco alcune delle sue caratteristiche:
- Progettato per gestire contemporaneamente centinaia di utenti (su un processore poco più potente di un vecchio 80386...)
- Memoria virtuale gestita col modello della segmentazione paginata.
- Capacità di gestire un sistema a multiprocessore con memoria condivisa.
- Nel 2007 Ken Thompson, a proposito del MULTICS osservò che era:

“...overdesigned and overbuilt and over everything. It was close to unusable”

Fase 4: il sistema MULTICS

- Fu sviluppato in **PL/1**, un linguaggio progettato nel 1964 alla IBM che doveva riunire le caratteristiche migliori del FORTRAN, COBOL e ALGOL. Ma il PL/1 era pesante e inefficiente, ebbe poca diffusione e non contribuì certo al successo del MULTICS.
- Il MULTICS offriva ambienti di sviluppo per molti linguaggi: PL/1, FORTAN, COBOL, LISP, a cui se ne aggiunsero altri col tempo. In quegli anni una tale versatilità era veramente notevole.
- Non c'è dubbio che lo sviluppo del MULTICS costituì il più grande progetto di ingegnerizzazione software del suo tempo, e portò alla scrittura di codice estremamente efficiente e compatto.
- Il sistema era progettato per rimanere attivo 24 ore su 24: componenti hardware (incluse le CPU!) potevano essere aggiunte o tolte dal sistema mentre questo continuava a girare.
50

Fase 4: il sistema MULTICS

- Il MULTICS fu sviluppato su un processore della General Elettrics a 36 bit con indirizzi a 18 bit, e il mainframe su cui era installato, un GE-645, era dotato di 2 megabyte di memoria primaria, una quantità enorme per quegli anni.
- E il kernel del MULTICS ne occupava permanentemente 135 Kbyte (il che era fonte di scherno per il mondo informatico del tempo). Ma in totale il MULTICS era composto di circa 300.000 righe di codice e occupava in tutto 4,5 Mbyte di memoria.
- In figura una reunion di multicians. Sulle magliette, davanti e dietro, sono stampate le scritte:
Save Honeywell, Buy a Multics e
Save Multics, Buy Honeywell



Fase 4: il file system moderno

- Il MULTICS non fu un successo, ma ebbe comunque un ruolo importante nella storia dei sistemi operativi per almeno due ragioni: ispirò (anche attraverso i suoi difetti) lo sviluppo dello Unix (di cui ci occuperemo fra poco), e all'interno del suo progetto fu proposta per la prima volta l'idea un file system gerarchico, soluzione poi adottata da tutti i sistemi futuri.
- Nel CTSS lo spazio per i file era sostanzialmente organizzato in modo che ogni utente potesse tenere i propri file su un nastro diverso, e in più era disponibile una sorta di directory comune per condividere i file.
- Si trattava dunque di una soluzione a directory singola, e ogni utente non poteva avere due file con lo stesso nome.

Fase 4: il file system moderno

- Ma man mano che gli utenti usavano i primi sistemi interattivi, emergevano chiare alcune esigenze nell'uso delle informazioni di cui aveva bisogno ciascun utente mentre interagiva col computer.
- La prima ovviamente era quella di poter organizzare i propri file in un modo più strutturato che non mettere tutto in un unico luogo, mischiando insieme informazione di tipo diverso.
- La seconda era di poter definire delle politiche chiare di accesso e uso dei file: i propri e quegli degli altri. Chi poteva fare, e cosa, con i file e l'informazione in essi contenuta?
- E poi i file dovevano essere facilmente accessibili e la loro integrità doveva essere garantita da una sessione di lavoro alla successiva⁵³

Fase 4: il file system moderno

- Nel 1965 **Robert Daley** e **Peter Neumann**, che facevano parte del gruppo che lavorava al MULTICS pubblicano un articolo che contiene l'essenza di ciò che ancora oggi chiamiamo **File System Gerarchico**.
- Interessante notare che l'articolo del 1965 costituiva solo una proposta, poiché l'implementazione effettiva fu portata a termine solo qualche anno dopo, e fu preceduta da quella del sistema **Titan**.
- Nella loro proposta Daley e Neumann osservano innanzi tutto che l'uso del file system dovrà essere indipendente da come viene implementato su una specifica macchina, e gli utenti useranno i file solo attraverso nomi simbolici: **tutto il lavoro di mappatura dei file sui dispositivi di memorizzazione dovrà essere invisibile all'utente.**

Fase 4: il file system moderno

- I file dovranno essere organizzati in una struttura ad albero i cui nodi interni costituiscono **directory** che possono contenere file e **sotto directory**, in una struttura gerarchica espandibile in cui nasce il concetto di **pathname di un file**.
- L'utente potrà decidere chi ha accesso a quali porzioni della propria gerarchia di file, e potrà scegliere di condividere alcune porzioni della propria gerarchia con opportuni **link**.
- Gli utenti avranno a disposizione **comandi per manipolare i file** e rimodellare la propria porzione di file system.
- Il sistema si incaricherà di garantire la privatezza e la sicurezza dei dati da usi impropri e malfunzionamenti del sistema.

Fase 4: il file system moderno

- Il sistema **Titan**, successore dell'Atlas, nel 1967 venne dotato di un sistema operativo time sharing (il **Titan Supervisor**) che includeva il primo file system moderno operativo.
- Era dotato di un hard disk da 128 Mbyte e memorizzava migliaia di file appartenenti a circa 700 utenti (26 dei quali potevano lavorare contemporaneamente on-line).
- I file venivano memorizzati sul disco in blocchi non contigui da 4 Kbyte mediante la tecnica della **File Allocation Table**, e venivano copiati automaticamente su nastro per aumentarne l'integrità.
- Il Titan fu anche il primo sistema a memorizzare in un file di sistema le password degli utenti in forma criptata, in modo da evitarne il facile furto.

Fase 4: il sistema operativo OS/360

- Il sistema operativo **OS/360** è degno di nota non certo per i suoi aspetti innovativi (quasi inesistenti), ma perché negli anni '60 dominò la scena a livello commerciale.
- All'inizio degli anni '60 la IBM produceva ormai un gran numero di mainframe e periferiche, in tutta la gamma di prezzi e prestazioni. Ma i sistemi di fascia alta e bassa erano incompatibili fra loro, cosicché era impossibile condividere i programmi e le applicazioni.
- I vertici IBM convenirono che queste incompatibilità aumentavano i costi del progetto e sviluppo di hardware e software, e riducevano le vendite perché i clienti evitavano di passare ai nuovi modelli. (in figura, l'IBM 1401, circa 1960).



Fase 4: il sistema operativo OS/360

- Dunque, nel 1964 la IBM lancia la linea di mainframe **System/360**, una gamma di computer di potenza e costi diversi per le diverse esigenze, ma con periferiche e software compatibili.
- Indubbiamente, pochi prodotti hanno avuto un impatto innovativo sull'economia, sul modo di percepire il lavoro e sulla diffusione dei computer come la serie 360, grazie proprio alla loro estrema compatibilità.
- Nei piani della IBM tutti i diversi modelli avrebbero dovuto usare un **unico sistema operativo, l'OS/360**, ma in effetti ne furono rilasciate più versioni leggermente diverse che si adattavano meglio alle caratteristiche hardware dei vari modelli.



Fase 4: il sistema operativo OS/360

- Sebbene l'idea di sistema time sharing fosse ormai nota, l'OS/360 era ancora un sistema multiprogrammato: il suo scopo fondamentale era di ridurre i tempi di attesa dovuti all'uso delle periferiche.
- L'IBM in effetti aveva sviluppato dei prototipi di sistema operativo time sharing, ma troppo tardi per installarli nei mainframe 360. Ci riprovò nel 1967 col **TSS/360**, che però si rivelò troppo inaffidabile.
- Fu rilasciato invece un prototipo alternativo, il **CP/67** (ma senza garanzia) che si sarebbe poi evoluto nel sistema timesharing **VM/370** della serie di mainframe 370 presentata nel 1972.
- Poiché doveva garantire la compatibilità del software su macchine diverse, lo sviluppo dell'OS/360 richiese un enorme sforzo progettuale, e gettò le basi del **software engineering** moderno: l'applicazione di principi scientifici allo sviluppo del software.⁵⁹

Fase 4: il sistema operativo Unix

- La realizzazione del sistema Unix è uno degli eventi cruciali dell'intera storia dell'informatica, e nessun altro sistema operativo ha avuto tanta influenza come lo Unix.
- A metà degli anni '80 Unix era ormai diventato lo standard di riferimento per i sistemi operativi timesharing, più di 100,000 sistemi Unix erano operativi, e praticamente qualsiasi università del mondo usava il sistema Unix (diffondendone così ulteriormente la conoscenza tra le nuove generazioni di programmatore e sistemisti)
- Addirittura, nel 2000 **Per Brinch Hansen** ha osservato che l'uso di

“un sistema superiore come lo Unix appare spesso così naturale da non invogliare i programmatore a cercarne uno migliore. Dopo trent'anni, si può ritenere che l'enorme diffusione dello Unix sia divenuto un ostacolo per ulteriori progressi.”

Fase 4: il sistema operativo Unix

- In effetti, è stato osservato che lo Unix ebbe la fortuna (o il merito) di comparire sulla scena al momento giusto, poco dopo la comparsa sul mercato dei primi minicomputer (soprattutto la serie **PDP** della **Digital**), sistemi hardware più piccoli ed economici dei mainframe.
- Gli utenti di questi nuovi computer erano assai insoddisfatti dei sistemi operativi disponibili (come abbiamo appena visto, la scena era dominata da un sistema ancora solo multiprogrammato, l'OS/360), ed erano quindi pronti per lo Unix.
- La maggior parte delle idee contenute nello Unix non erano effettivamente nuove ma, come ha osservato uno dei padri del sistema, **Dennis Ritchie**, costituivano

“una buona ingegnerizzazione di idee in giro già da un po’ in forme diverse, ed erano ora pronte per essere usate in modo conveniente.”⁶¹

Fase 4: il sistema operativo Unix

- Dunque come abbiamo visto, nella seconda metà degli anni '60 i Bell Labs erano impegnati nello sviluppo del MULTICS, progetto dal quale si ritirarono nel 1969.
- **Ken Thompson**, che aveva lavorato al MULTICS trovo in uno dei laboratori un PDP-7 poco usato, e decise di svilupparci sopra un sistema più facile da usare del MULTICS.
- Raggiunto da **Dennis Ritchie**, in pochi mesi i due svilupparono un file system gerarchico, un interprete dei comandi, un editor, e un gestore dei processi.
- In figura, Thompson (seduto) e Ritchie alla console del PDP-11 (fotografia di Peter Amer)



Fase 4: il sistema operativo Unix

- La prima versione del sistema era monoutente, ma lo Unix fu sviluppato fin dall'inizio come sistema interattivo multiutente.
- Il successo, tra i colleghi dei Bell Labs, della prima implementazione fece sì che Thompson e Ritchie potessero acquistare un più potente PDP-11/20 su cui proseguire lo sviluppo del sistema e aggiungere nuovi applicativi, in particolare per l'editing di testi.
- Nel 1970 **Peter Neumann** suggerisce di chiamare il nuovo sistema **UNICS** (**UNIplexed etc. etc**), come gioco di parole col MULTICS, e pare sia stato **Brian Kernighan** a proporre il definitivo **Unix**.
- In foto Thompson (sinistra) e Ritchie negli anni '80.



Fase 4: il sistema operativo Unix

- Nel 1973 lo Unix viene riscritto in C, linguaggio da poco messo a punto dallo stesso Ritchie proprio per questo scopo. **La nuova versione è un terzo più lunga di quella scritta direttamente in assembler, ma infinitamente più facile da capire, modificare e, soprattutto, installare su hardware diversi.**
- Lo stesso anno Ritchie e Thompson presentano lo Unix al *Symposium on Operating System Principles*, e come risultato ricevono le prime richieste per poter usare il sistema.
- Alla **AT&T** però era vietato entrare nel mercato dei computer, dunque lo Unix non poteva essere venduto come prodotto, ma poteva essere distribuito dietro licenza d'uso a chi ne facesse richiesta.
- Il codice sorgente viene spedito al solo costo del nastro e delle spese di spedizione, secondo la leggenda, con la firma “*Love, Ken*”. ⁶⁴

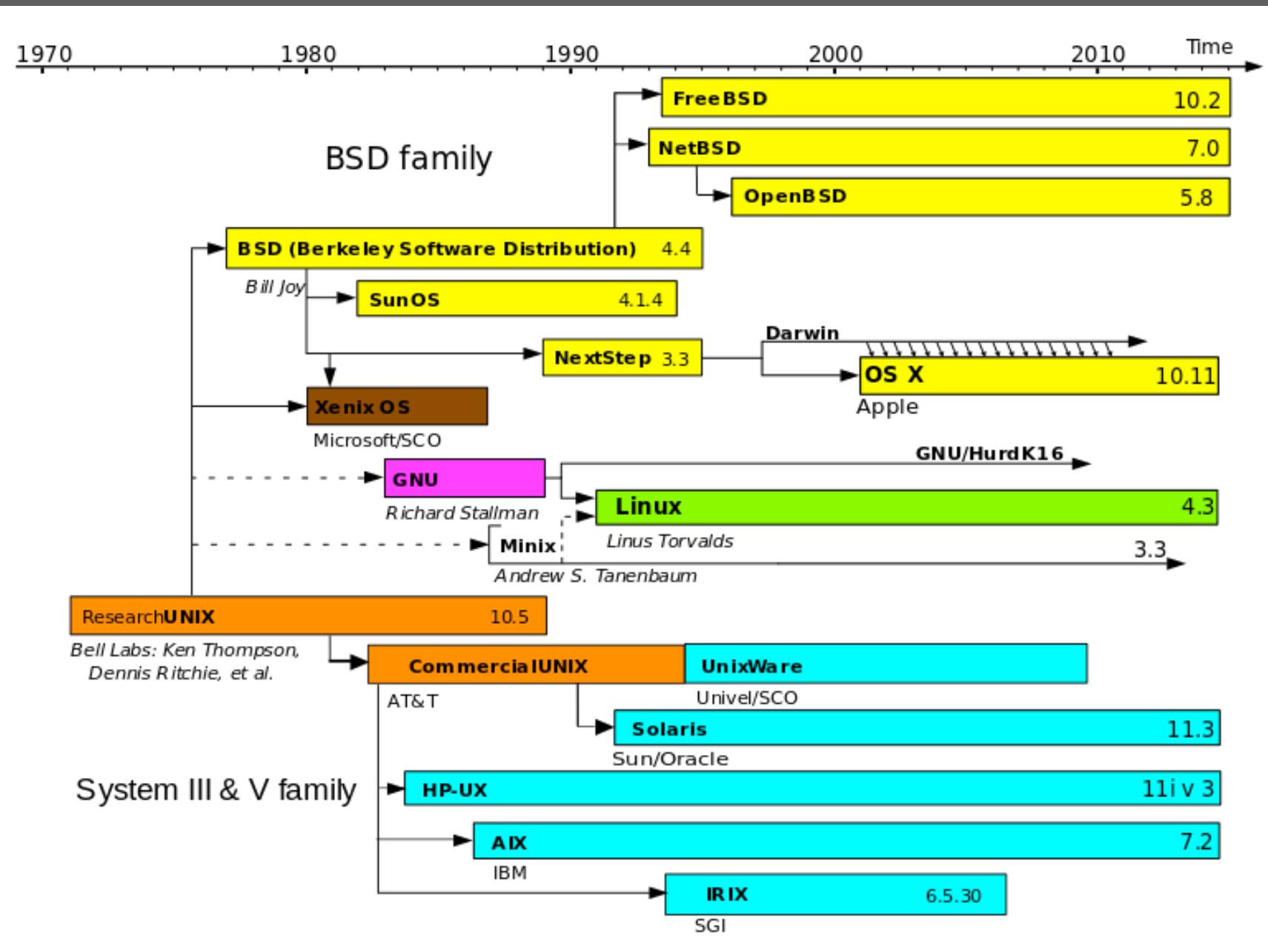
Fase 4: il sistema operativo Unix

- Nel 1974 pochi programmatori interessati allo Unix si trovano per la prima volta, e nel 1975 fondano la “*Unix User Group*”, ridenominata nel 1977 **USENIX** (la AT&T aveva proibito l’uso del termine Unix), e ancora oggi ampiamente attiva: www.usenix.org
- La USENIX ebbe un ruolo fondamentale nella diffusione, nello sviluppo e nel miglioramento dello Unix, grazie sia alla passione di coloro che ne facevano parte e sia alla disponibilità dei sorgenti.
- La AT&T cercò anche di guadagnare qualcosa dallo Unix, e nel 1977 rilascia lo **Unix V6**, con una licenza d’uso per attività non educational di circa 20,000 dollari.
- Ma ormai, molte versioni dello Unix erano già state sviluppate o in via di sviluppo, e nel 1978 si contavano più di 600 installazioni.

Fase 4: il sistema operativo Unix

- Nel lucido successivo vediamo l’evoluzione della famiglia dei sistemi Unix-like dalle origini fino al 2013 (fonte: Wikipedia).
- Unix-like significa che questi sistemi sono più o meno (ma sempre molto) compatibili con le specifiche Unix ufficiali, e Dennis Ritchie ha osservato che i sistemi Unix-like sono di fatto dei sistemi Unix.
- La famiglia è piuttosto ampia, e comprende sia prodotti free e open source ispirati allo Unix originale, e sia prodotti commerciali che hanno cambiato nome al sistema, come Sun OS, Solaris e OS X.
- Le due sotto-famiglie più importanti sono la **BSD** e la **System V**, sviluppati rispettivamente a Berkeley e alla AT&T a partire dal lavoro originale di Thompson e Ritchie

Fase 4: il sistema operativo Unix



Fase 4: il sistema operativo Unix

- Nel 1974 Ritchie e Thompson pubblicano sulle **Communication of the ACM** uno degli articoli più letti e citati di tutta la storia dell'informatica: **The Unix Time-Sharing System**.
- L'articolo contiene la quintessenza di ciò che normalmente intendiamo per sistema operativo moderno. Più di quarant'anni dopo, il contenuto di questo articolo è ancora perfettamente aggiornato.
- Ovviamente però, allo Unix originale sono state aggiunte molte altre cose, come ad esempio system call dedicate per la comunicazione e la sincronizzazione fra processi locali e remoti.
- Ritchie e Thompson non dimenticano di citare da dove hanno preso molte delle idee implementate nello Unix. Ad esempio, dal MULTICS la struttura del file system e il funzionamento della **Shell**, e da un sistema timesharing del 1965 sviluppato a Berkeley la **fork**.

Fase 5: sistemi concorrenti

- “Già a metà degli anni ’60 i sistemi operativi avevano raggiunto un livello di complessità ben aldilà dell’umana comprensione”
(P. B. Hansen)
- Come risultato, non solo i sistemi soffrivano spesso di thrashing, ma deadlock e starvation erano all’ordine del giorno. Così, i successivi dieci anni videro un grosso lavoro per sviluppare le basi concettuali necessarie a rendere i sistemi più comprensibili, prevedibili e affidabili.
- Questo lavoro pionieristico portò alla scoperta dei **principi fondamentali di programmazione concorrente**, e alla definizione di diversi modelli di strumenti software necessari ad implementare questi principi (tra cui ovviamente *semafori*, *regioni critiche condizionali*, e *monitor*)

Fase 5: il sistema THE

- Nel 1968 **Edger Dijkstra** sviluppa il sistema **THE** (Technische Hogeschool Eindhoven). È in effetti un sistema molto semplice, non è nemmeno timesharing, ma semplicemente multiprogrammato.
- Il sistema THE ha però due caratteristiche innovative fondamentali: è costruito a **strati sovrapposti**, che trasformano la macchina fisica in una **macchina virtuale** molto più facile da usare, e usa i **semafori** per la sincronizzazione e la comunicazione fra i processi.
- A proposito del progetto, Dijkstra osserverà che “*il livello intellettuale necessario per progettare un sistema software è fortemente sottostimato.*” Questa verità è di solito ignorata, il che spiega bene la qualità scadente del software che ci tocca spesso usare...
(in figura, Edger Dijkstra)



Fase 5: il sistema RC 4000

- Il sistema multiprogrammato **RC 4000** (noto anche come **Monitor**) del 1969 di **Per Brinch Hansen** è anch'esso un sistema sperimentale e non particolarmente efficiente (anzi...)
- Non era nemmeno un sistema operativo completo, ma il primo esperimento di sviluppo di un **kernel** (inizialmente chiamato *Nucleus*) attorno al quale sviluppare un sistema completo in base alle proprie esigenze.
- Questa è naturalmente al giorno d'oggi una tecnica scontata, ma nel 1969 era completamente nuova. Hansen scriveva:

“il sistema non fa assunzioni sul tipo di scheduling e sull’allocazione delle risorse: permette ad un programma (sic!) di iniziare altri in modo gerarchico. Così, il sistema fornisce un contesto generale per diverse strategie di scheduling: batch, interattive, real time, etc!”

Fase 5: il Concurrent Pascal

- Per Brinch Hansen è anche il padre del primo linguaggio di programmazione concorrente. Il **Concurrent Pascal**. del 1973. Si trattava di una estensione del Pascal contenente primitive per la sincronizzazione e la comunicazione tra processi.
- L'idea del Concurrent Pascal era che usando un linguaggio strutturato e ad alto livello sarebbe stato più facile scrivere programmi concorrenti liberi da errori in fase di esecuzione.
- La potenzialità del linguaggio fu mostrata con lo sviluppo del sistema operativo **Solo**, il cui intero codice era così sintetico da poter essere pubblicato completamente su una rivista di informatica del tempo: a disposizione dei lettori, facile da comprendere e da modificare.
- Oggi i linguaggi concorrenti sono molti, e molti linguaggi general purpose (come Ada e Java) includono costrutti per la concorrenza.⁷²

Fase 6: sistemi per personal computer

- Tendiamo ad associare i personal computer e i loro sistemi operativi ai sistemi Windows e OSX. Ma questi sono prodotti commerciali che hanno usato (anche migliorando) idee venute ben prima di loro.
- Come abbiamo visto, lo sviluppo della tecnologia informatica permise, tra la fine degli anni 60 e l'inizio degli anni '70, la costruzione di piccoli computer dai costi molto contenuti ma anche dalla potenza di calcolo piuttosto limitata.
- Naturalmente, per questi semplici computer era impossibile fare girare i sistemi operativi che erano stati sviluppati per architetture ben più potenti: il codice del kernel avrebbe occupato più spazio della poca memoria primaria disponibile.
- Nasceva dunque l'esigenza di, ma anche l'opportunità per, sperimentare nuove soluzioni software.

Fase 6: il sistema OS 6

- Se dobbiamo trovare un capostipite dei sistemi operativi per personal computer questo è probabilmente il sistema **OS 6**, sviluppato tra il 1969 e il 1972 da **Joe Stoy** e **Christopher Strachey** alla Oxford University.
- Scritto in BCPL (che abbiamo già incontrato a proposito delle origini del C), era in grado di far girare un programma alla volta su un piccolo computer dotato di soli 32 Kbyte di memoria principale.
- In effetti, l'OS 6 ha una importanza storica principalmente per l'implementazione del file system e del meccanismo di input/output del programma in esecuzione. Entrambi furono adottati identicamente nello sviluppo del sistema Xerox Alto, disponibile a partire dal 1973.

Fase 6: Alto/Pilot/Star Xerox Systems

- Abbiamo già incontrato nella parte sulle architetture la macchina **Alto** progettata alla Xerox, dotata di uno schermo bit-mapped, mouse e interfaccia Ethernet.
- Il sistema operativo Alto, sviluppato completamente in BCPL tra il 1973 e il 1976, era di una semplicità estrema: eseguiva un processo alla volta in modalità strettamente sequenziale: personal computing ai minimi termini.
- Il sistema Alto è rimasto famoso soprattutto per la sua rivoluzionaria interfaccia grafica: è importante notare che però era legata alla applicazione, e non era parte del sistema operativo.



Fase 6: Alto/Pilot/Star Xerox Systems

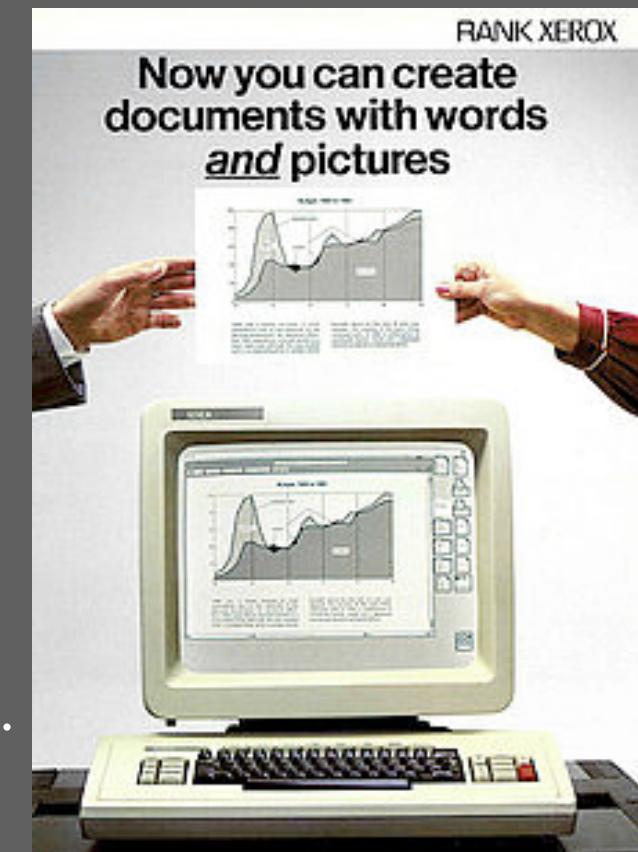
- All'Alto seguono nel 1980 il **Pilot**, multiprogrammato, e poi il **Cedar**, e nel 1981 lo **Xerox Star**, il primo sistema dotato di mouse e interfaccia a finestre ad essere lanciato sul mercato.
- In effetti, nel 1981 i progettisti dello Star, sfruttando le idee dell'Alto già da due anni lavoravano all'idea di **ufficio elettronico**:

“decidemmo di creare la controparte elettronica degli oggetti di un ufficio: carta, classificatori, caselle della posta, calcolatrici: una metafora elettronica dell’ufficio fisico, in modo che l’ambiente elettronico sembrasse più familiare e facile da apprendere.

In Star i documenti non sono rappresentati solo da nomi di file su disco, ma da immagini sullo schermo. Possono essere selezionati puntandoli col mouse e cliccando su uno dei sui tasti. Aperti, appaiono sullo schermo esattamente come verranno poi stampati”⁷⁶

Fase 6: Alto/Pilot/Star Xerox Systems

- Naturalmente, oggi tutto ciò ci è estremamente familiare, dunque forse è difficile apprezzare come, agli inizi degli anni '80, quelle idee risultassero rivoluzionarie.
- Possiamo sicuramente inserire le interfacce grafiche come una delle innovazioni più importanti nella tecnologia dei sistemi operativi.
- In figura, la pubblicità di uno **Xerox Star Dandelion**, in commercio dall'aprile 1981 al costo di 16,595 \$ (circa 45,000 \$ attuali)
- Lo Star non fu però un successo commerciale: costava troppo, era molto lento, la Xerox non aveva credito come produttore di computer, e le strategie di marketing furono sbagliate. In tutto ne furono venduti circa 25.000 esemplari.



Fase 6: CP/M → MS-DOS → Windows

- A volte il successo di una ricerca scientifica o di una impresa commerciale dipendono anche dal caso: sicuramente questo è vero per la storia della Microsoft e dei sistemi operativi DOS/Windows.
- A volte poi, la fortuna di qualcuno (diciamo, **Bill Gates**) corrisponde alla sfortuna di qualcun altro (poniamo, **Gary Kildall**). Naturalmente, hanno un ruolo fondamentale anche le scelte personali, che solo a posteriori possono rivelarsi giuste o terribilmente sbagliate...
- Nei primi anni ‘70 Gary Kildall era un giovane e talentuoso ricercatore informatico che amava sperimentare in campi diversi. Acquistò tra le altre cose una cpu 4004 per cui scrisse alcuni programmi, e si mise a collaborare come consulente con la Intel.



Fase 6: CP/M → MS-DOS → Windows

- Nel 1973 mette a punto un linguaggio di programmazione ad alto livello, il PL/M, e soprattutto il **CP/M** (Control Program/**M**onitor, poi rinominato “for **M**icroprocessor”), un semplice sistema operativo per controllare il drive di un floppy disk da una CPU Intel 8080.
- Nel 1974, con la moglie, fonda la (**I**ntergalactic) **D**igital **R**esearch per commercializzare il CP/M, che apparentemente non aveva riscosso molto successo presso la Intel.
- Il CP/M viene inizialmente pubblicizzato su riviste di elettronica e informatica per hobbisti, e poiché poteva girare su macchine con configurazioni hardware diverse, divenne presto, e quasi inaspettatamente, popolarissimo: una sorta di standard *de facto* per i personal computer che andavano diffondendosi in quegli anni.
79

Fase 6: CP/M → MS-DOS → Windows

- All'apice della sua popolarità il CP/M poteva girare su migliaia di modelli diversi di computer, e la Digital research in pochissimi anni arrivò a fatturare milioni di dollari.
- Dunque il CP/M fu il primo sistema operativo per personal computer a larghissima diffusione. Come mai? Perché Gary Kildall fu l'ideatore, nel 1975, del **BIOS** (**B**asic **I**nput **O**utput **S**ystem), la logica che permette di interfacciare un sistema operativo con lo specifico hardware su cui il sistema sta girando.
- Prima del BIOS, un sistema operativo doveva essere scritto pensando ad una ben precisa piattaforma hardware, limitandone così la portabilità.

(In figura una pubblicità del CP/M del 1978)



Fase 6: CP/M → MS-DOS → Windows

- Un sistema operativo per personal computer doveva occupare pochissimo spazio e faceva ben poche cose: recuperare un file dal disco e mandarlo in esecuzione, e permettere una manipolazione di base dei file, spostarli, copiarli, rimuoverli.
- Il CP/M era suddiviso in tre parti:
 - **CCP** (**C**onsole **C**ommand **P**rocessor): un interprete dei comandi, per impartire comandi come: *copia il file XXX sul file YYY*
 - **BDOS** (**B**asic **D**isk **O**perating **S**ystem): un elenco di 40 routine di base necessarie ad implementare il comandi del CCP, caricate ad indirizzi prestabiliti in RAM
 - **BIOS** (**B**asic **I**nput/**O**utput **S**ystem): che conteneva le funzioni di base *machine dependent*, come il posizionamento delle testine del disco e la lettura di uno specifico settore.

Fase 6: CP/M → MS-DOS → Windows

- Dunque, un comando come: *copia il file XXX sul file YYY* veniva scomposto dal CCP in istruzioni di base del BDOS come:
 - 1) *trova il file XXX,*
 - 2) *crea il file YYY,*
 - 3) *trasferisci il contenuto di XXX in YYY.*
- Queste istruzioni BDOS venivano a loro volta trasformate in istruzioni BIOS come:
 - 1) *seleziona il floppy disk specificato (a volte ne erano disponibili 2)*
 - 2) *sposta la testina di lettura sul settore interessato*
 - 3) *trasferisci i byte letti in RAM*
- E così via.

Fase 6: CP/M → MS-DOS → Windows

- Questa progettazione *a strati* del sistema operativo è considerata ai giorni nostri ovvia, ma negli anni ‘70, e soprattutto nel contesto dei primi personal computer, era assolutamente innovativa.
- Portare il CP/M da una piattaforma hardware ad un’altra (che usassero la stessa CPU) richiedeva la sola riscrittura del BIOS, riducendo enormemente i tempi di diffusione del sistema operativo.
- Dunque, programmi scritti (di solito in BASIC) per il CP/M erano facilmente portabili su macchine diverse che usassero la stessa CPU: come conseguenza, un’enorme quantità di software fu scritto per girare nell’ambiente CP/M.

Fase 6: CP/M → MS-DOS → Windows

- **Wordstar** (uno dei primi wordprocessor per PC), **dBase** (tra i primi database per PC) furono scritti per il CP/M. Per gli amanti delle memorabilia, in questa pagina in stile retrò trovate tutto sul CP/M: www.classiccmp.org/cpmarchives/
- Il 4 aprile del 1975 **Paul Allen** e **Bill Gates** fondano la **Micro-Soft**, e scrivono per il CP/M un compilatore BASIC derivato dall'Altair BASIC. (disegnato personalmente da Allen e Gates, in figura il primo logo Micro-Soft, che poco dopo verrà rinominata **Microsoft**)
- Alla fine degli anni '70 la IBM decide di entrare nel mercato dei personal computer e naturalmente, vista la sua enorme diffusione, la scelta del sistema operativo da adottare ricade sul CP/M.



Fase 6: CP/M → MS-DOS → Windows

- Il CP/M girava persino sugli Apple II, grazie ad una scheda di espansione prodotta su progetto Microsoft, per cui la IBM contatta Gates pensando che sia in possesso della licenza d'uso del CP/M.
- La Microsoft però non aveva i diritti d'uso del sistema, e Bill Gates indirizza la IBM a **Gary Kildall**, col quale, nell'estate del 1980, viene organizzato un incontro.
- Qui la storia si fa confusa, e anche se in rete si trovano molti dettagli, rimane poco chiaro perché non fu possibile stipulare un accordo che permettesse alla IBM di usare il CP/M sui suoi PC di prossima commercializzazione. (ma sta di fatto che negli anni a venire Kildall verrà indicato come “*the man who could have been Bill Gates*”)

Fase 6: CP/M → MS-DOS → Windows

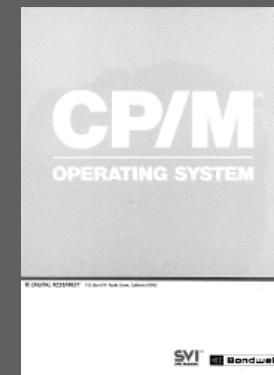
- I dirigenti IBM allora tornano da Bill Gates chiedendogli di trovare una soluzione, e questi non si lascia scappare l'occasione:
 - compra dalla **Seattle Computer Products** la licenza d'uso dell'**86-DOS**, una variante del CP/M per la CPU Intel 8086
 - Ingaggia il 24enne **Tim Paterson**, lo sviluppatore dell'86-DOS, per portare il sistema su processori 8088, che saranno usati nei PC IBM.
 - Rinomina il nuovo software **MS-DOS** e ne vende la licenza d'uso alla IBM, che lo commercializzerà sui suoi PC con la sigla **PC-DOS**
 - L'accordo tra IBM e Microsoft permette inoltre a quest'ultima di vendere il DOS ad altre compagnie, produttrici di PC compatibili IBM.

(In figura, Tim Paterson, nel 1986)



Fase 6: CP/M → MS-DOS → Windows

- Naturalmente, non appena Kildall entra in possesso di una copia del PC-DOS realizza che è così simile al CP/M da violare la legge sulla proprietà intellettuale del software (legge per altro a quel tempo ancora poco chiara) e minaccia di fare causa alla IBM.
- Per evitare problemi, la IBM accetta di offrire i suoi PC con il sistema CP/M in alternativa al PC-DOS, ma ad un costo 6 volte superiore, e con molto meno supporto e applicativi compatibili. Ovviamente, tutti finiranno per continuare ad usare il PC-DOS...



Fase 6: CP/M → MS-DOS → Windows

- Nel 1985 la Microsoft rende disponibile la prima versione di una interfaccia grafica per l'MS-DOS che chiama **Windows**: si tratta di un applicativo che può essere lanciato all'interno del DOS e trasforma lo schermo del PC in un desktop “moderno”.
- Attraverso diverse versioni, Windows diventa un sistema operativo completo nel 1990 con la versione 3.0, a cui faranno seguito nel 1992 la versione 3.1 e nel 1993 la **NT**, multi-processore e multiutente.
- Nel 1995 viene lanciato, con una gran campagna pubblicitaria, **Windows95**, che segna definitivamente la fine dell'era DOS.



Fase 6: DOS → Mac OS → OS X

- Alla Apple hanno sempre mantenuto una certa aria di sufficienza nei confronti della concorrenza (ricordate la pubblicità fatta in occasione del lancio sul mercato dei PC IBM).
- E dunque ecco alcuni degli slogan con logo Apple apparsi dopo il lancio di Windows95 su diverse riviste, e anche venduti come stickers:



Introducing Windows 95

It lets you use more than eight characters to name your files.

It has a trash can you can open and take things out of again.

It lets you drop files anywhere you want on the desktop.

Imagine that.

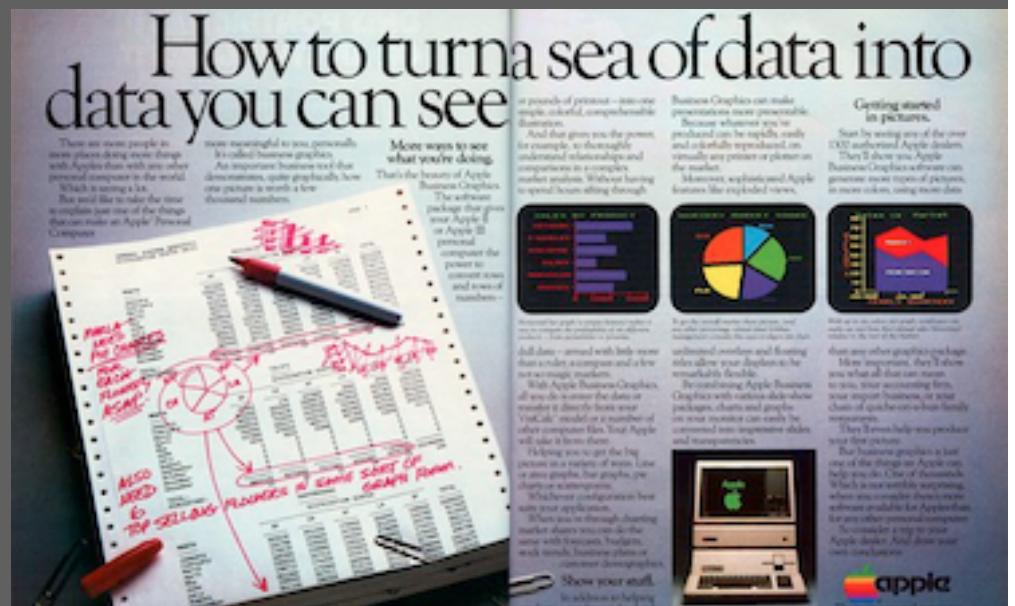


Fase 6: DOS → Mac OS → OS X

- Tendiamo ad associare alla Apple i celeberrimi MAC OS e OS X, ma il loro primo computer venduto su larga scala, l'Apple II, del 1977, era dotato di un ben più umile **DOS**.
- Abbiamo già incontrato questo acronimo: **Disk Operating System**, un generico software che permetteva di gestire la memoria secondaria di un computer e organizzarne i file memorizzati in un file system.
- Il DOS era in sostanza il sistema operativo di un PC: ridotto all'essenziale, memorizzato su floppy disk o su ROM e caricato in RAM all'avvio del computer, permetteva di lanciare i programmi degli utenti e una manipolazione di base del file system.
- Tutte ispirate al CP/M, a cavallo tra gli anni '70 e '80 ne furono sviluppate più versioni, a seconda del tipo di macchina su cui giravano. Tra i più famosi: l'Atari-DOS e il Commodore-DOS.⁹⁰

Fase 6: DOS → Mac OS → OS X

- Naturalmente, sull'Apple II girava un **Apple-DOS**, sviluppato inizialmente da Steve Wozniak e alcuni suoi collaboratori.
- Una idea vincente della Apple fu di rendere pubbliche tutte le caratteristiche software della sua macchina. Ciò permetteva a programmatori terzi di sviluppare nuovi applicativi sfruttando a fondo tutte le potenzialità della macchina su cui dovevano girare.
- Ad esempio **VisiCalc**, il primo foglio elettronico disponibile sul mercato, fu esplicitamente sviluppato per l'Apple II. In tanti contesti aziendali l'uso di un tale applicativo era così fondamentale che era la necessità di usarlo che faceva vendere l'Apple II!



Fase 6: DOS → Mac OS → OS X

- All'inizio degli anni '80 però la Apple era ormai lanciata verso lo sviluppo di sistemi con interfaccia grafica, soprattutto dopo aver visto in azione alla Xerox, nel dicembre del 1979, il sistema Alto.
- Il primo tentativo, **Lisa**, entrò in commercio nel gennaio del 1983, ma fu un insuccesso: il sistema era lento, inaffidabile e costoso, e dopo un solo anno viene rimpiazzato, il 24 gennaio 1984, dal Celeberrimo **Macintosh** (più tardi noto come Macintosh **128k**).
- Il Macintosh era dotato del sistema operativo **System**, che dalla versione 7.5 cambierà nome in **Mac OS**. Il System/Mac OS costituiva indubbiamente una rivoluzione rispetto ai sistemi operativi a cui era abituato il grande pubblico, sostituendo completamente l'interfaccia da linea di comando con una a finestre.
- In soli due mesi ne furono venduti circa 70.000 Macintosh! 92

Fase 6: DOS → Mac OS → OS X

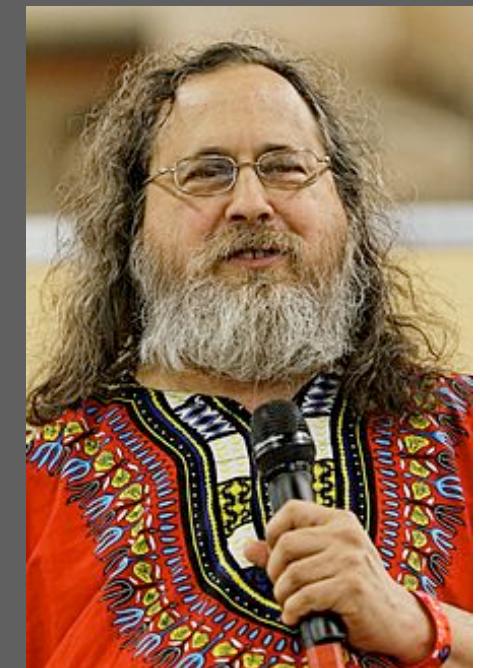
- Il sistema operativo Mac OS arriva fino alla versione 9, e col passaggio alla versione 10, nel 2001, il sistema cambia il proprio nome in **Mac OS X** (X essendo il numero romano per 10).
- Le versioni del Mac OS X vengono numerate a partire dalla 10. Così, il primo Mac OS X è il Mac OS X 10.0, a cui seguiranno il 10.1, e così via. A partire dalla versione 10.8, “*Mountain Lion*” del 2012, viene abbandonato il prefisso “Mac”.
- Il passaggio Mac OS / Mac OS X del 2001 non è però un semplice rebranding del nome, ma un cambiamento tecnologico radicale: i (Mac) OS X sono sistemi Unix-like derivati da **NEXTSTEP**, il sistema sviluppato da Jobs durante il suo esilio dalla Apple, tra il 1985 e il 1997.



Fase 6: GNU, Minix, Linux

- Naturalmente, non è possibile lasciare fuori da una storia dell'informatica il sistema **Linux**, e anche se lo inseriamo in questa sezione, la storia del Linux ha una portata che travalica quella dei sistemi operativi per personal computer.
- Come per altre situazioni che abbiamo già visto, il Linux sembra nascere quasi per caso, e in un modo che rasenta il mito. Tuttavia, questo sistema operativo non esisterebbe se prima del suo sviluppo non fossero successe alcune altre cose importanti.
- Nel 1983 **Richard Stallman**, al MIT di Boston, da il via al progetto **GNU**: un progetto collaborativo di massa per lo sviluppo di software libero.

(in figura, Stallman nel 2014, il quale ama definirsi “*l'ultimo degli ackers*”)



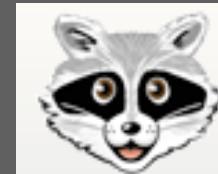
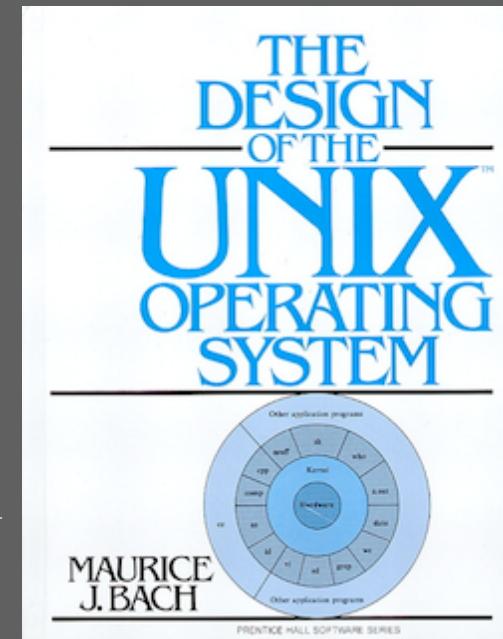
Fase 6: GNU, Minix, Linux

- L'obiettivo dichiarato del progetto GNU è di dare agli utilizzatori di computer la libertà di usare i loro computer attraverso **software che sia stato sviluppato in maniera collaborativa e che tutti siano liberi di usare, condividere, studiare e modificare.**
- Naturalmente, perché *tutto* il software di un sistema sia libero è necessario che anche la sua parte più importante, il sistema operativo e le sue utilities, sia free software.
- Stallman inizia dunque a lavorare ad un sistema operativo free, avendo in mente lo Unix come riferimento, e decide di chiamare il suo sistema GNU, un acronimo ricorsivo che sta per: **GNU is Not Unix**. Il nome verrà poi cambiato in **GNU Hurd** (o semplicemente **Hurd**, che è di nuovo un acronimo ricorsivo...)



Fase 6: GNU, Minix, Linux

- Lo sviluppo tuttavia procede molto a rilento, ancora all'inizio degli anni '90 un kernel Hurd non è disponibile...
- Nel 1985 viene immessa sul mercato la prima CPU a 32 bit della Intel, l'**80386**.
- Nel 1986 **Maurice Bach** pubblica il fondamentale: **The Design of the Unix Operating System**, la definitiva descrizione di come è implementato lo Unix.
- Nel 1987 **Andrew Tanenbaum** pubblica i sorgenti del **MINIX**, un sistema operativo Unix-like sviluppato per scopi didattici (accompagna il testo del Tanenbaum sui sistemi operativi)



MINIX 3

Fase 6: GNU, Minix, Linux

- Il MINIX però era un sistema a 16 bit, e il porting su un processore a 32 bit come il 386 non funzionava molto bene. E d'altra parte una licenza Unix per il 386 costava troppo per un privato.
- Di fatto, queste furono le motivazioni che spinsero **Linus Torvalds** a sviluppare un nuovo sistema operativo. Torvalds dichiarò che se in quegli anni fossero stati già disponibili il GNU Hurd o il **386BSD**, il Linux non sarebbe mai nato...
- Torvalds sviluppa il Linux nei primi mesi del 1991 sul suo PC dotato di processore 80386, usando il **GNU C Compiler** come linguaggio, il MINIX come ambiente di sviluppo, e il *Design of the Unix Operating System* come riferimento. (In foto, Torvalds nel 1991)



Fase 6: GNU, Minix, Linux

- Il 25 agosto 1991 Torvalds, 21 anni (!), posta sul NG *comp.os.minix* un messaggio che ora appartiene alla storia dell'informatica:

Hello everybody out there using minix

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

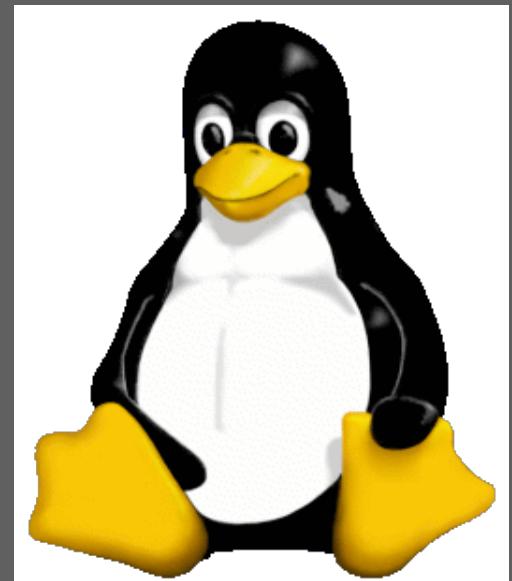
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Fase 6: GNU, Minix, Linux

- Il nome del sistema scelto da Torvalds era **Freak**, (*Freak*, con la *x* in riferimento allo Unix). I sorgenti del sistema furono caricati sul server FTP dell'università di Helsinki da un amico di Torvalds, il quale, senza nemmeno consultarlo, cambiò il nome del progetto in **Linux**.
- Nel 1992 il kernel Linux viene rilasciato sotto licenza **GNU GPL** (General Public Licence). Da questo momento, inizia lo sviluppo dei vari componenti che trasformano un kernel in un sistema operativo completo (shell, compilatori, interfacce grafiche, etc. etc.)
- Torvalds ha poi dichiarato: “*making Linux GPL'd was definitely the best thing I ever did*”
- Il logo Linux **Tux** (**Torvalds' Unix**) viene scelto nel 1996, quando Torvalds ricordò di essere stato morso da un pinguino durante una visita allo zoo di Camberra, in Australia.



Fase 6: GNU, Minix, Linux

- Linux gode immediatamente di un enorme successo, e la maggior parte dello sviluppo di Linux (il kernel e il software di contorno distribuito insieme al kernel) viene portato avanti dalle migliaia di programmatori della comunità Linux.
- Questi sviluppatori sono raggruppati sotto vari progetti e compagnie che forniscono varianti del sistema sotto vari nomi (**Fedora**, **RedHat**, **Ubuntu**, **SUSE**, etc. etc.).
- Linux è free software, ma molte compagnie (membri della **Linux Fundation**, associazione no-profit che promuove lo sviluppo del kernel Linux) investono nello sviluppo del Linux, e alcune di esse fanno pagare la loro distribuzione Linux in cambio di supporto tecnico e aggiornamenti garantiti.

Fase 7: Sistemi Distribuiti

- Un sistema operativo distribuito integra il funzionamento di un insieme di computer collegati fra loro in modo tale che ogni utente abbia l'illusione di stare usando una unica unità computazionale, che in realtà è costituita da un insieme di nodi.
- Così ad esempio, tutti gli utenti vedranno (e potranno accedere ai file di) un unico file system, il quale invece è fisicamente distribuito su diversi nodi del sistema.
- Gli utenti manderanno in esecuzione i programmi dalla loro postazione di lavoro, ma sarà il sistema operativo distribuito a decidere su quale nodo eseguire i programmi, cercando di bilanciare il carico di lavoro sull'intero sistema.
- Eventuali guasti di uno dei nodi del sistema saranno (per quanto possibile) mascherati dal sistema operativo, in modo tale da continuare ad offrire agli utenti i servizi di cui hanno bisogno.¹⁰¹

Fase 7: Sistemi Distribuiti

- Lungo gli anni ‘80 incominciarono a diffondersi le prime architetture distribuite, la cui configurazione era spesso quella di alcuni computer collegati da una rete locale Ethernet in grado di condividere alcune risorse, come stampanti e file system distribuito.
- I concetti fondamentali dei sistemi operativi erano ormai stati scoperti, e i primi sistemi distribuiti cominciavano a venire sviluppati, principalmente a partire dal sistema Unix.
- Nella maggior parte dei sistemi distribuiti, la comunicazione fra processi remoti era basata sul meccanismo delle **Remote Procedure Call (RPC)**: sostanzialmente una estensione dell’Inter Process Communication (IPC) a processi che girano su macchine diverse, che permetteva di invocare l’esecuzione di procedure remote rispetto al processo chiamante.

Fase 7: Sistemi Distribuiti

- Si possono trovare le radici delle RPC nel sistema RC4000 che abbiamo già incontrato nella sezione sui sistemi concorrenti. Ma una prima implementazione estesa delle RPC (e anche la creazione del nome) avvenne nel sistema Cedar dello Xerox Alto nel 1981.
- Fu la Sun a implementare le prime RPC in ambiente Unix, nel 1984, e costituirono la base per lo sviluppo del **Network File System (NFS)**, un file system distribuito su una rete di computer che viene visto dagli utenti come se fosse implementato su un'unica macchina.
- In effetti, i sistemi distribuiti non sono mai riusciti ad andare oltre una fase sperimentale, né si è formato uno standard accettato globalmente, e NFS è l'unico loro aspetto che ormai è diffuso e utilizzato a livello commerciale su tutti i principali sistemi operativi.

Fase 7: Sistemi Distribuiti

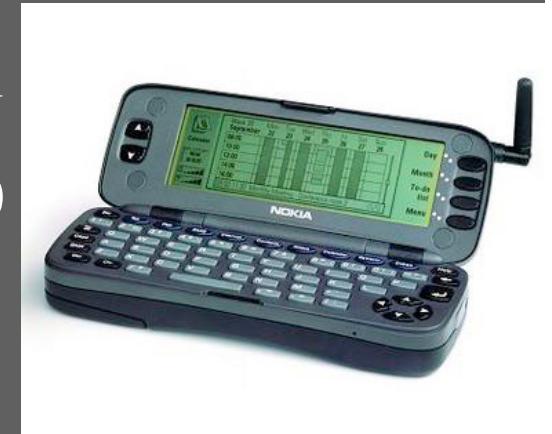
- A titolo esemplificativo possiamo dunque giusto menzionare due sistemi sperimentali che hanno dimostrato praticamente la possibilità di realizzare un vero sistema operativo distribuito.
- Lo **Unix United System** fu sviluppato nel 1982 all'università di Newcastle. Aggiungendo uno strato di software al di sopra di normali sistemi Unix, era in grado di trasformare cinque macchine PDP11 in un ambiente Unix omogeneo.
- I file system locali diventavano un unico file system, visto in ugual modo da tutti gli utenti, che potevano usare ogni risorsa remota (dischi, stampanti, nastri magnetici) come se fosse locale.
- Il sistema **Amoeba** fu sviluppato in Olanda negli anni '80 da Andrew Tanenbaum. Era formato da una rete di workstation single user e processori addizionali, con un file system distribuito due volte più veloce di NFS.

Fase 8: Sistemi per dispositivi mobili

- È orientativamente dalla metà degli anni 90' in poi che l'avanzamento tecnologico ha permesso di costruire dispositivi elettronici sofisticati portabili, collegati alla rete telefonica e dati, ed estremamente versatili: **smartphone** e **tablet PC**
- Proprio come per gli ordinari computer, l'uso semplice è intuitivo di questi dispositivi richiede la presenza di un sofisticato sistema operativo, che sia in grado di integrarne le tante funzioni, da quelle telefoniche, alla navigazione in rete, alla produzione e riproduzione di informazione multimediale, all'uso mediante touchscreen.
- Un sistema per dispositivi mobili non differisce molto da un classico sistema operativo, deve però garantire una qualche funzionalità *real time*, e deve essere capace di gestire al meglio le risorse di memoria e l'alimentazione a batteria.

Fase 8: Sistemi per dispositivi mobili

- I sistemi operativi per dispositivi mobili nascono con i sistemi sviluppati per i primi dispositivi *smart*, cellulari e palmari.
- Con il **Nokia 9000**, del 1996, la telefonia mobile entra nel campo dei computer. Dotato di una CPU derivata dal 386 e 8 Mbyte di RAM il Nokia 9000 aveva una tastiera *qwerty* e poteva inviare fax e connettersi a Internet visualizzando dati graficamente complessi.
- Il primo dispositivo palmare, dotato di calendario, rubrica, possibilità di prendere note e fare calcoli, viene sviluppato dalla Apple nel 1993. Il **MessagePad** ha un processore ARM e 8 Mbyte di RAM. Questi dispositivi erano anche noti come **PDA**: Personal Digital Assistant.



Fase 8: Sistemi per dispositivi mobili

- Dall'unione delle caratteristiche di questi due tipi di dispositivi nascono i veri e propri **smartphone**, ogni nuovo modello aggiungendo qualcosa ai precedenti (ad esempio l'**iPhone**, nel 2007, è stato il primo smartphone a permettere l'accesso ai social network).
- Nei primi anni 2000 compaiono sul mercato i primi **tablet PC**, sostanzialmente dei computer portatili molto sottili, con schermo molto più ampio di uno smartphone, connessi in rete, e pilotati mediante touchscreen (alcuni in realtà hanno vere e proprie tastiere)
- I sistemi operativi sviluppati per questi dispositivi sono molti: **Android**, **Apple iOS**, **Windows Mobile**, **Symbian** e **Blackberry** i più famosi. Tra questi, i primi due sono di gran lunga i più diffusi.

Fase 8: Android

- La **Android** viene fondata nel 2003, e nel 2005 viene acquistata da **Google**. Il sistema incomincia il suo sviluppo in questo periodo: è basato su kernel Linux, ma viene sviluppato in Java anziché in C.
- Android è open source e multitasking e può girare praticamente su qualsiasi dispositivo mobile. Ha fatto il suo esordio in commercio nel 2008, sullo smartphone **HTC Dream**.
- Le sue caratteristiche open source fanno sì che per esso siano sviluppate tantissime applicazioni indipendenti (ormai superano il milione) e così Android è divenuto il sistema più usato: copre tra il 65% e l'80% del mercato dei sistemi operativi per dispositivi mobili.



Fase 8: Apple iOS

- **Apple iOS** è stato rilasciato nel 2007, e ovviamente gira solo su dispositivi mobili Apple: iPhone e iPad.
- iOS usa un kernel **XNU** (**X** is **N**ot **U**nix), originariamente sviluppato per il NextSTEP di Steve Jobs. XNU è un ibrido tra BSD Unix e il (micro) kernel **Mach** sviluppato alla Carnegie Mellon University.
- Ad Apple iOS spetta tra il 15% e il 25% del mercato, sommato alla fetta assai più cospicua di Android, questo ci dice quanto siano poco diffusi gli altri sistemi operativi per dispositivi mobili...



Diffusione dei sistemi operativi

- Quanto sono diffusi i diversi sistemi operativi attualmente in uso? Compilare delle statistiche è molto difficile, soprattutto perché i dati grezzi non sono sempre disponibili in modo preciso, e cambiano abbastanza rapidamente.
- Nel campo degli smartphone domina Android, seguito a distanza da iOS, mentre nei desktop/Laptop sono le varie versioni di Windows a farla da padrone (ma naturalmente non è possibile tenere traccia di coloro che comprano un PC e poi ci installano sopra Linux).
- Il mercato dei Web Server è diviso più o meno alla pari tra Linux, Unix e Windows. Nei server (non Web) dominano Unix e Linux, e nei supercomputer (sebbene ovviamente il loro numero sia estremamente limitato) viene principalmente usato Linux, in qualche versione adattata alla particolare installazione.

Diffusione dei sistemi operativi

- Ecco una tabella riassuntiva, i valori sono solo orientativi, specialmente i decimali... (derivata da meta-fonti Wikipedia):

Tipo di dispositivo	Linux	Unix-like	Windows
Desktop/laptop	1,65%	9,57% (OSX)	88,77%
Smartphone/ Tablet	64,89% (Android)	23,56% (iOS)	1,70%
Web Server	36,72%	30,18%	33,10%
Supercomputer	99,79% (custom)		
Server	28%	72%	