Sistema operativo

Un esempio di sistema operativo con interfaccia grafica Unity: Ubuntu GNU/Linux

Un **sistema operativo** (abbreviato in **SO**, in lingua inglese **OS**, "operating system"), in informatica, è un insieme di componenti software, che consente l'utilizzo di varie apparecchiature informatiche (ad esempio di un computer) da parte di un utente.

Esempi di sistemi operativi sono il DOS, Windows, Unix, le distribuzioni GNU/Linux, Mac OS, Android, iOS.

Caratteristiche generali

Esso garantisce l'operatività di base di un calcolatore, coordinando e gestendo le risorse hardware di processamento (processore) e memorizzazione (memoria primaria), le periferiche, le risorse/attività software (processi) e facendo da interfaccia con l'utente, senza il quale quindi non sarebbe possibile l'utilizzo del computer stesso e dei programmi/software specifici, come applicazioni o librerie software.

È dunque un componente essenziale del sistema di elaborazione che funge da "intermediario" tra l'utente e la macchina ed inoltre è una base al quale si appoggiano gli altri software, che dunque dovranno essere progettati e realizzati in modo da essere riconosciuti e supportati da quel particolare sistema operativo. Assieme al processore, con cui è strettamente legato, costituisce la cosiddetta piattaforma del sistema di elaborazione.

Seguendo un po' l'evoluzione storica dei sistemi operativi, in generale un sistema operativo può essere:

- *monoutente*, se un solo utente per volta può accedere alle risorse dell'elaboratore;
- *multiutente*, se più utenti possono accedere alle risorse dell'elaboratore che a sua volta può essere:
  - *seriale*, sequenzialmente uno per volta;
  - *parallelo*, ciascuno parallelamente agli altri;
- *monotasking*, se in grado di eseguire un solo compito o task (processo) alla volta:
- *multitasking*, se in grado di svolgere più compiti parallelamente attraverso una certa politica di scheduling (es. timesharing).
- *portabile* o meno su differenti architetture hardware di processori.

Funzioni principali

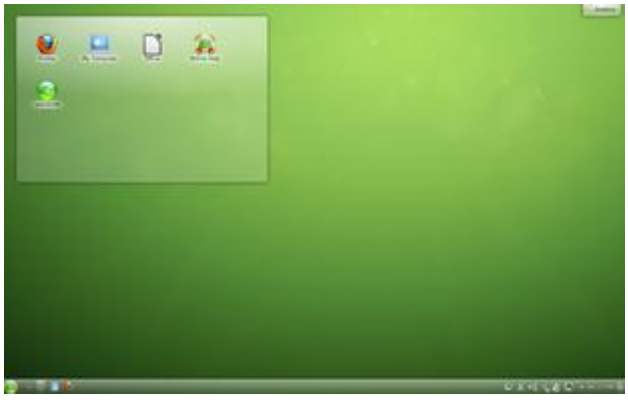
Secondo una definizione più rigorosa, il sistema operativo è un insieme di subroutine e strutture dati responsabili:

- del controllo e della gestione delle componenti hardware che costituiscono il computer (processi di Input/Output da e verso le periferiche collegate al sistema)
- dell'esecuzione dei programmi (processi) che su di esso vengono eseguiti, assegnando ad essi le necessarie risorse per l'avanzamento dei processi.

Se il sistema di elaborazione prevede la possibilità di memorizzazione aggiuntiva dei dati su memoria di massa, come accade nei computer general purpose, esso ha anche il compito di:

- gestire l'archiviazione e l'accesso ai file. I programmi possono gestire l'archiviazione dei dati su memoria di massa (ottenendo strutture complesse, come un database), servendosi delle procedure messe a disposizione del sistema operativo. La componente del SO che si occupa di tutto ciò viene chiamata file system.

Infine, se è prevista interazione con l'utente, viene solitamente utilizzata allo scopo un'interfaccia software (grafica o testuale) per accedere alle risorse hardware (dischi, memoria, I/O in generale) del sistema. D'altra parte, un sistema operativo può essere utilizzato anche su una macchina che non preveda interazione diretta con un essere umano (per un esempio, vedi smart card o determinati sistemi embedded) spesso dunque più leggero e semplificato.



Un altro esempio di sistema operativo (basato su kernel Linux): openSUSE

Solitamente un sistema operativo installato su computer fornisce anche degli applicativi di base per svolgere elaborazioni di diverso tipo.

Sebbene molte delle funzionalità sopraddette non siano spesso immediatamente visibili/percepibili dall'utente, l'importanza del sistema operativo di un calcolatore è cruciale: oltre alla necessità di gestione delle funzionalità di base sopraddette, al di là delle prestazioni massime offerte dall'hardware dell'elaboratore stesso, il sistema operativo determina di fatto efficienza e buona parte delle prestazioni effettive di funzionamento dell'intero sistema ad esempio in termini di latenze di processamento, stabilità, interruzioni o crash di sistema.

Struttura

Un generico sistema operativo moderno si compone di alcune parti standard, più o meno ben definite.

Il **kernel**
: un gruppo di funzioni fondamentali, strettamente interconnesse fra loro e con l'hardware, che vengono eseguite con il privilegio massimo disponibile sulla macchina ossia in modalità kernel; il kernel fornisce le funzionalità di base per tutte le altre componenti del sistema operativo, che assolvono le loro funzioni servendosi dei servizi che esso offre. A seconda del tipo di sistema operativo il kernel può inglobare altre parti (kernel *classico*, monolitico o modulare) o fornire solo funzioni base delegando più funzioni possibile a oggetti/gestori esterni (microkernel).

Un sistema di gestione di memoria primaria
: Esso è eseguito dal MMU(Memory Management Unit) che alloca la memoria primaria richiesta dai programmi e dal sistema operativo stesso, salva sulla memoria di massa le zone di memoria temporaneamente non usate dai programmi (memoria virtuale) e garantisce che le pagine *swappate* vengano riportate in memoria se richieste.

Uno scheduler
: che scandisce il tempo di esecuzione dei vari processi e assicura che ciascuno di essi venga eseguito per il tempo richiesto. Normalmente lo scheduler gestisce anche lo *stato* dei processi e può sospenderne l'esecuzione nel caso questi siano in attesa senza fare nulla, assegnando le risorse inutilizzate ad altri processi che le necessitano (esempio classico è la richiesta di dati da disco). Nei sistemi operativi *realtime* lo scheduler si occupa anche di garantire una *timeline*, cioè un tempo massimo di completamento per ciascun task in esecuzione, ed è notevolmente più complesso.

Il gestore di file system
: si occupa di esaudire le richieste di accesso alle memorie di massa. Viene utilizzato ogni volta che si accede a un file sul disco, e oltre a fornire i dati richiesti tiene traccia dei file aperti, dei permessi di accesso ai file. Inoltre si occupa anche e soprattutto dell'astrazione logica dei dati memorizzati sul computer (directory, ecc).

Uno spooler
: che riceve dai programmi i dati da stampare e li stampa in successione, permettendo ai programmi di proseguire senza dover attendere la fine del processo di stampa.

Un'interfaccia utente (shell)
: che permette agli utenti di interagire con la macchina.

A seconda dei casi, un particolare sistema operativo può avere tutti questi componenti o solo alcuni. Una ulteriore differenza fra i sistemi operativi è data dal tipo di comunicazione fra le varie componenti: i sistemi operativi classici sono basati su chiamate dirette di funzioni, mentre molti sistemi operativi moderni, soprattutto quelli che adottano microkernel, si basano sul *message passing*, sullo scambio di messaggi fra le loro varie parti e fra il sistema operativo e i programmi che fa girare.

Vediamo ora alcune classi di sistemi operativi possibili, dal più semplice al più complesso.

Shell

Il secondo passo verso una migliore gestione del computer si ha con lo sviluppo di una interfaccia utente separata dal kernel, un interprete di comandi che funzioni anche da interfaccia utente ovvero da Shell. Questa shell primitiva di solito funge anche da interprete per un linguaggio di programmazione: a seconda delle scelte dei progettisti del software può essere un vero linguaggio oppure un più semplice linguaggio di scripting con cui creare comandi batch. Era il tipico sistema operativo degli home computer degli anni 80, come il Commodore 64 e il Sinclair ZX Spectrum.

DOS

Un computer diventa molto più utile ed efficace se dotato di una memoria di massa: per gestirla serve un gestore di file system, cioè un software che in sintesi è composto da un insieme di funzioni che permetta di organizzare e gestire (accesso o lettura, scrittura o memorizzazione, ordinamento) i dati sulla superficie dei mezzi di memorizzazione secondo una struttura ben precisa. I sistemi operativi che risiedevano su disco e capaci di gestire un file system sono detti genericamente Disk Operating System, cioè DOS appunto. L'esemplare più famoso è senz'altro l'MS-DOS di Microsoft, oggi sostituito dall'interfaccia grafica Windows, ma che era alla base dei sistemi operativi Windows 95/98/Me. Ne esiste anche una versione libera compatibile con i suoi programmi, il FreeDOS, ed altre versioni come il DR-DOS.

Scheduler

I programmi non hanno sempre realmente bisogno della CPU: a volte, invece di eseguire istruzioni, stanno aspettando che arrivino dei dati da un file, o che l'utente prema un tasto della tastiera. Quindi si può, in linea di principio, usare questi tempi "morti" per far eseguire un altro programma. Quest'idea, sorta fin dai primi anni cinquanta, si concretizzò nei *sistemi operativi multitasking*, cioè dotati di uno scheduler che manda in esecuzione più processi (esecuzione di programmi), assegnando a turno la CPU a ognuno e sospendendo l'esecuzione dei programmi in attesa di un evento esterno (lettura/scrittura sulle memorie di massa, stampa, input utente ecc.) finché questo non si verifica. Dovendo ospitare in memoria centrale più programmi nello stesso tempo, i sistemi multitask hanno bisogno di più memoria rispetto a quelli monotask: perciò questo tipo di sistemi operativi è quasi sempre dotato di un gestore di memoria virtuale. Inoltre, con più programmi simultaneamente attivi, il controllo delle risorse hardware diventa una reale necessità e non è più possibile farne a meno. Esistono sostanzialmente due modi di implementare il multitasking: *cooperative* e *preemptive* multitasking.

Nel primo sono i programmi che, spontaneamente, cedono il controllo al sistema non appena hanno terminato la singola operazione in corso; nel secondo è lo scheduler che ferma i programmi allo scadere del tempo assegnato e trasferisce il controllo dall'uno all'altro:

- il *cooperative multitasking* assorbe meno risorse di calcolo e non ha quasi ritardo di commutazione per il cambio di task, e inoltre non richiede nessuna struttura hardware dedicata, il che rende possibile implementarlo su qualunque calcolatore; per contro è molto vulnerabile a errori nei programmi (in genere il crash di un programma fa cadere l'intero sistema) e l'isolamento fra processi è molto debole. È il modello usato dai vecchi sistemi.
- il *preemptive multitasking* necessita di CPU che implementino in hardware sia dei livelli di privilegio per l'esecuzione del codice, sia una logica specifica per il context switch, il cambio di task eseguito dallo scheduler. Poiché l'interruzione dei programmi è arbitraria, al cambio di task il sistema operativo è costretto a salvare tutti o quasi i registri della CPU e ricaricarli con quelli salvati dal task che subentra, perdendo molto tempo. A fronte di queste

maggiori richieste, il preemptive multitasking offre una sicurezza del sistema molto maggiore e una virtuale immunità ai crash di sistema causati da errori nei programmi. È il modello usato dai moderni sistemi operativi.

Multitasking

Se un computer può far girare più programmi contemporaneamente, allora può anche accettare comandi da più utenti contemporaneamente: in effetti dal multitasking alla multiutenza o timesharing il passo è tecnicamente molto breve, ma fa sorgere una serie di nuovi problemi dal punto di vista della sicurezza dei sistemi operativi: come distinguere i vari utenti tra loro, come accertarsi che nessun utente possa causare danni agli altri o alla macchina che sta usando. Questi problemi di sicurezza informatica si risolvono assegnando un account univoco per ciascun utente, assegnando un proprietario ai file ed ai programmi e gestendo un sistema di permessi per l'accesso ad essi, e prevedendo una gerarchia di utenti (cioè di account) per cui il sistema rifiuterà tutti i comandi potenzialmente "pericolosi" e li accetterà soltanto se impartiti da un utente in cima alla gerarchia, che è l'amministratore del sistema.

Sistema operativo *online*

Mediante opportuni software, il sistema operativo può avere la funzionalità di *desktop remoto*, che consente tramite una connessione internet l'accesso al proprio elaboratore e a tutti gli applicativi e dati salvati in uno specifico momento. Tramite accesso remoto sono possibili le stesse operazioni che si possono fare stando davanti al proprio elaboratore. L'accesso è protetto da *user* e *password* che possono essere al limite condivisi da una comunità di utenti. In questo caso, il desktop remoto è un'evoluzione della tradizionale cartella condivisa. La cartella condivisa già permetteva la comunicazione di qualsiasi file, dunque anche di eseguibili, installabili da remoto in locale, oltreché di dati.

Sistemi operativi realtime

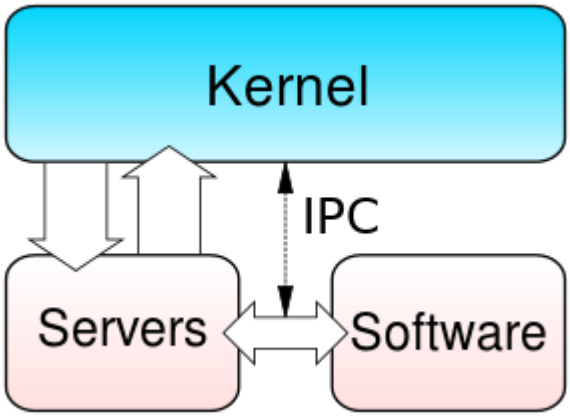
Un sistema operativo realtime è un particolare tipo di sistema operativo, in grado di garantire una risposta entro un dato tempo limite (millisecondi o microsecondi) a qualunque evento esterno. Questo requisito porta a una diversa struttura del sistema: per esempio i sistemi realtime usano spesso il polling (meno efficiente, ma deterministico) invece degli interrupt per gestire le periferiche, e non hanno memoria virtuale.

I sistemi realtime si trovano spesso in ambito industriale, musicale o comunque dove sia necessario ottenere una risposta dal sistema in un tempo massimo prefissato. A loro volta i sistemi realtime si possono dividere in due categorie: hard e soft, a seconda dei tempi di risposta; un PC che faccia girare un gioco in 3D, per esempio, può essere considerato un sistema "soft-realtime".

Parti del sistema operativo

Kernel

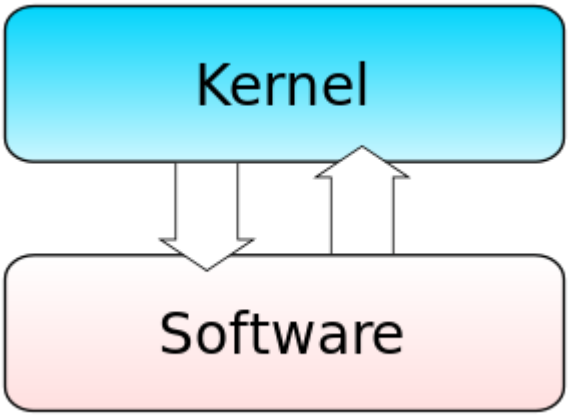
Il kernel è il motore di un sistema operativo. Si tratta di un software che ha il compito di fornire ai moduli che compongono il sistema operativo e ai programmi in esecuzione sul computer le funzioni fondamentali ed un accesso controllato all'hardware, sollevandoli dai dettagli della sua gestione.



Schema di Microkernel

Quali funzioni sia opportuno che il kernel debba fornire e quali possano essere demandate a moduli esterni è oggetto di opinioni divergenti: se il kernel di un sistema operativo implementa soltanto un numero molto ristretto di funzioni, delegando il resto ad altre parti esterne dette *server* o *moduli* in comunicazione con il kernel stesso, si parla di **microkernel**. Il vantaggio di un siffatto sistema operativo è la maggiore semplicità del suo kernel, del suo sviluppo, la possibilità di cambiare facilmente i moduli ed una certa tolleranza ai guasti in quanto se un modulo "crolla" (*crash*) tutto il sistema rimane "normale; lo svantaggio è invece l'interazione più complessa e costosa fra kernel e le altre componenti del SO stesso, che spesso rallenta il sistema e/o lo rende meno stabile.

Un kernel tradizionale, ovvero **monolitico** e altri, integra invece dentro di sé la gestione della memoria virtuale, lo scheduler e i gestori di file system, nonché i driver necessari per il controllo di tutte le periferiche collegate. Questo tipo di kernel è più complesso da progettare, mantenere ed aggiornare, ma è anche più veloce, efficiente e stabile. Una sua evoluzione è costituita dai kernel "*modulari*", che mantengono al loro interno lo scheduler e i gestori di file system e memoria virtuale ma separano alcune funzioni non essenziali in moduli a sé stanti, da caricare in memoria solo in caso di effettivo uso della funzione o periferica di loro competenza.



Schema di kernel monolitico

Sulla distinzione fra microkernel e kernel monolitico di notevole interesse è il famoso dibattito fra Torvalds e Tanenbaum "LINUX is obsolete" che potete trovare su comp.os.minix al seguente collegamento.

File system

Il file system è il modo in cui i file sono immagazzinati e organizzati su un dispositivo di archiviazione, come un hard disk o un CD-ROM. Esistono molti tipi di file system, creati per diversi sistemi operativi, per diverse unità di memorizzazione e per diversi usi. Si possono identificare due grandi classi di file system: quelli per unità locali, destinate ad organizzare fisicamente i dati su un disco, e i file system *distribuiti*, nati per condividere i dati fra più computer collegati attraverso una rete, superando le differenze fra sistemi operativi e file system locali delle varie macchine.

File system per unità locali

File system

- Amiga FileSystem
- Caching File System (CFS)
- Ext2
- Ext3
- Ext4
- File Allocation Table (FAT)
- FAT 32
- HFS
- HFS Plus
- HPFS
- ISO 9660
- Journaled File System (JFS)
- Minix
- NTFS
- ReiserFS
- Unix File System (UFS)
- XFS
- ZFS
- Swap

File system distribuiti

- Network File System (NFS)
- Coda
- Andrew file system (AFS)

Scheduler

Lo scheduler è il componente fondamentale dei sistemi operativi multitasking, cioè quelli in grado di eseguire più processi contemporaneamente (va inteso simbolicamente. In realtà non c'è una gestione parallela dei processi ma avviene in sequenza, i tempi sono talmente brevi che all'utente sembrerà che i programmi vadano contemporaneamente. Lo scheduler si occupa di fare avanzare un processo interrompendone temporaneamente un altro, realizzando così un cambiamento di contesto (*context switch*). Generalmente computer con un processore sono in grado di eseguire un programma per volta, quindi per poter far convivere più task è necessario usare lo scheduler. Esistono vari algoritmi di scheduling che permettono di scegliere nella maniera più efficiente possibile quale processo far proseguire.

Gestione input/output e periferiche

La gestione dell'input/output ovvero delle periferiche di sistema è attuata attraverso il meccanismo dell'interrupt da parte delle periferiche stesse che chiamano in causa il sistema operativo il quale opererà un cambiamento di contesto (context switch) all'interno del ciclo del processore assegnando al processore il compito di input/output richiesto. Un'altra modalità tipica di gestione delle periferiche, alternativa agli interrupt, è il polling.

Gestore di memoria

Il *gestore di memoria* è la componente del sistema operativo che si occupa di gestire ed assegnare la memoria primaria ai processi che ne fanno richiesta immediatamente prima dell'elaborazione. La gestione della memoria è necessaria anche per tenere traccia di quanta memoria è impegnata e di quanta invece è disponibile per soddisfare nuove richieste: in mancanza di un sistema di gestione, si avrebbe prima o poi il caso nefasto di processi che ne sovrascrivono altri, con gli ovvi inconvenienti.

Un altro buon motivo per registrare la memoria usata dai vari processi è il fatto che, in caso di errori gravi, i processi possono andare in *crash* e non essere più in grado di comunicare al sistema che la memoria che occupano può essere liberata: in questo caso è compito del gestore di memoria, dopo la terminazione anomala del processo, marcare come libere le zone di memoria possedute dal processo "defunto", rendendole disponibili per nuove allocazioni.

Per poter gestire i programmi, divenuti processi, è necessario che tutti gli indirizzi definiti in essi siano calcolati in forma relativa alla prima istruzione del programma (come se il programma dovesse essere caricato a partire dall'indirizzo 0 di memoria centrale). Al momento del caricamento, che può essere eseguito in qualsiasi zona libera della memoria, gli indirizzi relativi verranno sommati al primo indirizzo di effettivo caricamento, diventando così assoluti: INDIRIZZO ASSOLUTO = INDIRIZZO RELATIVO + INDIRIZZO DI PARTENZA.

Una modalità/meccanismo tipico di gestione/assegnazione della memoria ai programmi/processi da parte del sistema operativo è il paging.

Gestore di memoria virtuale

Nel caso il sistema disponga di un meccanismo di memoria virtuale, il gestore della memoria si occupa anche di *mappare* (indirizzare) la memoria virtuale offerta ai programmi sulla memoria fisica e sui dischi rigidi del sistema, copiando da memoria a disco rigido e viceversa le parti di memoria necessarie di volta in volta ai programmi, senza che i programmi stessi o gli utenti debbano preoccuparsi di nulla.

Protezione della memoria

La protezione della memoria è un sistema per prevenire la corruzione della memoria di un processo da parte di un altro. Di solito è gestito via hardware (ad esempio con una MMU, Memory management unit) e dal sistema operativo per allocare spazi di memoria distinti a processi differenti.

Interfaccia utente

Si tratta di un programma che permette all'utente di interagire con il computer. Esistono sostanzialmente due famiglie di interfacce utente: interfaccia a riga di comando e interfacce grafiche (GUI) come ad esempio il desktop.

Android: Un esempio di sistema operativo mobile basato sul kernel Linux e sviluppato da Google per Smartphone e Tablet

Spooler di stampa

Lo **spooler di stampa** è stato storicamente il primo modulo esterno del sistema operativo ad essere implementato, per risolvere il problema della gestione delle stampe su carta. Infatti, essendo le stampanti elettromeccaniche dei dispositivi molto lenti, i primi programmi per elaboratore dovevano necessariamente sprecare molto tempo di CPU, estremamente prezioso all'epoca, per controllare la stampante ed inviarle i dati. Quindi venne ideato un programma separato, che girava con una priorità molto bassa e che era visto dagli altri programmi come una normale stampante: in realtà invece lo spooler accumulava i dati che un programma doveva stampare in una apposita area di memoria RAM, e poi si faceva carico del processo di stampa vero e proprio lasciando gli altri programmi liberi di continuare la loro esecuzione.

Il meccanismo fondamentale dello spooler di stampa è rimasto sostanzialmente invariato dai suoi albori fino ad oggi: con gli anni e con il progredire della tecnologia le modifiche più rilevanti sono state la capacità di gestire più stampanti selezionabili a piacere, e la capacità di gestire anche

stampanti remote, collegate cioè non direttamente al computer su cui gira lo spooler ma ad altri elaboratori connessi via rete.

I sistemi distribuiti in rete

Tra le varie ipotesi d'uso di un sistema operativo c'è anche la possibilità di gestione di un sistema distribuito in rete. In tal caso la computazione viene distribuita tra più computer collegati in rete tra loro. In questo modo le risorse e il carico computazionale vengono condivise e bilanciate, ottenendo una maggiore affidabilità e costi più contenuti nella scalabilità. Una configurazione funzionalmente simmetrica permette che tutte le macchine componenti abbiano lo stesso ruolo nel funzionamento del sistema e lo stesso grado di autonomia. Un'approssimazione pratica di questa configurazione è il clustering. Il sistema viene suddiviso in cluster semiautonomi, dove ognuno di essi, a sua volta, è costituito da un insieme di macchine e da un server cluster dedicato.

Installazione e avvio

Tipicamente il sistema operativo, una volta installato sulla macchina, risiede nell'hard disk pronto ad essere caricato nella RAM durante la fase di avvio della macchina.

Dual boot

Tipicamente più sistemi operativi possono essere installati sulla stessa macchina in modalità dual boot, selezionando poi il sistema desiderato nella fase di avvio del PC attraverso il boot manager. Tutto ciò è possibile solo in virtù dell'operazione di partizionamento della memoria secondaria (hard disk) in più settori logici indipendenti dove ciascuno può ospitare un diverso sistema.

Avvio

All'accensione del computer il BIOS, dopo la fase di POST, esegue nella cosiddetta fase di boot, attraverso il boot loader, il caricamento del kernel del sistema operativo dall'hard disk alla RAM, come qualunque programma, pronto ad essere eseguito dal processore, rendendo la macchina pronta all'uso da parte dell'utente. Nel caso di sistemi operativi ad interazione con l'utente questa fase, dopo il login iniziale da parte dell'utente stesso, tipicamente comporta anche il caricamento di tutte le impostazioni di configurazione (*settings*) e profilo utente.