LEZIONE 3

GESTIONE DEI PROCESSI

**Differenza tra programma e processo:** un **programma** è una serie di istruzioni non eseguibili, quando questo viene mandato in esecuzione si parla di **processo**. Un processo per svolgere i propri compiti necessita di alcune risorse (CPU, memoria, file e dispositivi I/O). **RICORDA BENE** Ad un programma caricato in una macchina per essere eseguito possono corrispondere possono corrispondere più processi.

* Il SO è responsabile delle seguenti attività connesse alla gestione dei processi:

1. **Creazione e cancellazione** dei processi utente e di sistema
2. **Sospensione e ripristino** dei processi
3. Fornitura di meccanismi per:

-**sincronizzazione** dei processi -**comunicazione** tra processi

Gestione della memoria centrale La RAM è un vasto vettore di dimensioni che variano tra le centinaia di migliaia e i miliardi di parole, ciascuna delle quali è dotata di un proprio **indirizzo** (perché bisogna conoscere la locazione di un dato per sapere dove andarlo a cercare in caso servisse). La RAM è un magazzino di dati velocemente accessibile ed è condivisa dalla CPU e da alcuni dispositivi di I/O.

* **La RAM è volatile**, e perde le informazioni in caso di guasto di sistema.
* Il SO è responsabile delle seguenti attività connesse alla gestione della RAM:

1. **Tenere traccia di quali parti della memoria sono attualmente usate e da chi o che cosa**
2. **Decidere quali processi si debbano caricare nella memoria quando vi sia spazio disponibile**
3. **Assegnare e revocare lo spazio di memoria secondo le necessità**

Gestione di file

* Un file è una raccolta di informazioni correlate, definite dal loro creatore. Comunemente i file rappresentano programmi (sia sorgente, sia oggetto) e dati
* Il SO è responsabile delle seguenti attività connesse alla gestione di file:

1. **Creazione e cancellazione** di file
2. **Creazione e cancellazione** di directory  **DIRECTORY:** sono delle cartelle che contengono gruppi di file
3. Fornitura delle funzioni fondamentali per la gestione di file e directory
4. Associazione dei file ai dispositivi di memoria secondaria
5. Creazione di copie di riserva (backup) dei file su dispositivi di memorizzazione non volatili

Gestione del sistema di I/O Il sistema di I/O è composto delle seguenti parti:

* **Un sistema di buffer-caching**
* Un’interfaccia generale per i driver dei dispositivi
* I driver per gli specifici dispositivi

**DRIVER:** Il driver è un pezzo di software che serve a mettere in comunicazione il sistema con uno specifico dispositivo. Si parla di driver perché i costruttori di dispositivi (schede video, stampanti ecc) in qualche modo, hanno bisogno di far funzionare la propria produzione sulle macchine, quindi devono fornire oltre l’hardware, anche delle regole che devono essere capaci di far colloquiare la loro risorsa con il resto del sistema.

Gestione della memoria secondaria (hard disk)

* Poiché la RAM è troppo piccola per contenere tutti i dati e tutti i programmi, e il suo contenuto va perduto se il sistema si spegne, il calcolatore deve disporre di una **memoria secondaria** a sostegno della memoria centrale.
* **La maggior parte dei moderni sistemi di calcolo impiega i dischi come principale mezzo di memorizzazione secondaria, sia per i programmi che per i dati**
* Il SO è responsabile delle seguenti attività connesse alla gestione dei dischi:

1. **Gestione dello spazio libero**
2. **Assegnazione dello spazio**
3. **Scheduling del disco**

Reti (sistemi distribuiti)

* Un **sistema distribuito** è un insieme di unità di elaborazione che **NON condividono la memoria, i dispositivi periferici o un clock;** ciascun processore dispone di una propria memoria locale e di un suo clock.
* **Le unità d’elaborazione sono collegate da una rete di comunicazione**
* La comunicazione avviene utilizzando un **protocollo (regole di comunicazione)**
* Un sistema distribuito offre all’utente l’accesso alle varie risorse di sistema
* L’accesso a una risorsa condivisa permette di:

1. Accelerare il calcolo
2. Aumentare la disponibilità dei dati
3. Incrementare l’affidabilità

Sistemi di protezione

* La **protezione** è definita da ogni meccanismo che controlla l’accesso da parte di programmi, processi o utenti alle risorse di un sistema di calcolo
* Il meccanismo di protezione deve:

1. Distinguere tra uso autorizzato e non autorizzato
2. Specificare i controlli che devono essere attivati
3. Fornire strumenti di miglioramento dell’affidabilità

Interprete dei comandi E’ un programma che una volta eseguito serve ad avere un colloquio diretto con la macchina mediante un’interfaccia

* Molti comandi si impartiscono al SO attraverso **istruzioni di controllo** che riguardano:

1. Creazione e gestione di processi
2. I/O
3. Gestione della memoria secondaria
4. Gestione della RAM
5. Accesso al file-system
6. Protezione
7. Reti

Servizi di un SO

* **Esecuzione di un programma:** capacità del sistema di caricare un programma nella memoria ed eseguirlo
* **Operazioni di I/O:** poiché i programmi utenti non possono eseguire direttamente operazioni di I/O, il sistema operativo deve offrire mezzi adeguati
* **Gestione del file-system:** capacità del programma di leggere, scrivere, creare e cancellare file
* **Comunicazioni:** scambio di informazioni tra processi in esecuzione nello stesso calcolatore e tra processi in esecuzione in calcolatori diversi collegati per mezzo di una rete. La comunicazione si può realizzare tramite una **memoria condivisa** o attraverso lo **scambio di messaggi**
* **Rilevamento d’errori:** capacità di rilevare eventuali errori che possono verificarsi nella CPU, nei dispositivi di memoria, nei dispositivi di I/O e nei programmi utenti.

Funzioni addizionali di un SO Esiste anche un’altra serie di funzioni del SO che non riguarda direttamente gli utenti, ma assicura il funzionamento efficiente del sistema stesso.

* **Assegnazione delle risorse:** se sono in corso più sessioni di lavoro di utenti o sono contemporaneamente in esecuzione più processi, il SO provvede all’assegnazione delle risorse necessarie a ciascuno di essi
* **Contabilizzazione dell’uso delle risorse:** registrazione degli utenti che usano il calcolatore, con segnalazione di quali e quante risorse vengono impiegate, a fini di addebito dei costi di preparazione di statistiche (ovvero si gestisce l’utente valutandone l’uso delle risorse che loro effettuano. Ciò è molto importante perché si riesce a contabilizzare l’uso di molte cose (es. quanta stampa l’utente ha fatto)
* **Protezione:** assicura che l’accesso alle risorse del sistema sia controllato

System call Le system call permettono di colloquiare con i livelli più bassi del sistema. In pratica una system call è una chiamata che un’applicazione, in maniera diretta o indiretta, effettua alle funzioni di livello più basso (scrittura su un disco, accessi alla memoria ecc)

* **Passaggio dei parametri:** ogni volta che si effettua una chiamata, questa deve dipendere da chi ha fatto la chiamata, quindi deve contenere al proprio interno una serie di parametri d’input da trattare. **Questi parametri sono passati in diversi modi:**  -Passare i parametri in **registri** -Memorizzare i parametri in un blocco o tabella di memoria e **passare l’indirizzo del blocco**, in forma di parametro, in un registro -Collocare (**push**) i parametri in una pila da cui sono prelevati (**pop**) dai SO

**Passaggio di parametri in forma di tabella**

Immagine che contiene screenshot

Descrizione generata automaticamente

Tipi di chiamate Le system call vengono classificate per categoria: -Controllo dei **processi** -Gestione dei **file** -Gestione dei **dispositivi** -Gestione delle **informazioni** -**Comunicazione**

Esecuzione dell’MS-DOS (Microsoft disk-operative-system) MS-DOS era il primo sistema a larga diffusione del SO; era molto snello/piccolo molto semplice e veloce nella sua esecuzione. MS-DOS girava su un disco ed era organizzato cosi:

Immagine che contiene screenshot

Descrizione generata automaticamente **Spiegazione** **FIGURA (a):** Si carica il nucleo, subito dopo viene lanciato l’interprete dei comandi e poi un comando di qualche applicazione che sarà lanciata nello spazio libero di memoria. **FIGURA (b)**: Dopo aver lanciato l’esecuzione, si avrà sempre il nucleo in basso, l’interprete dei comandi sopra il kernel, però sopra in una parte dello spazio libero di memoria, ci sarà caricato il processo, di conseguenza lo spazio libero sarà ridotto.

Immagine che contiene screenshot

Descrizione generata automaticamenteEsecuzione di più programmi nel SO UNIX

**Spiegazione** Molto più **versatile** del MS-DOS, in quanto l’interprete dei comandi può essere posizionato in qualsiasi parte della memoria. Un’altra differenza è che mentr3e nel MS-DOS la memoria libera è addensata tutta nella stessa parte, qui può trovarsi in più punti. (infatti notare che sopra il nuclei ci sta un prfocesso,poi ‘interprete dei comandi e cosi via, ecco perché è versatile)

Modelli di comunicazione La comunicazione può avvenire attraverso il modello **scambio di messaggi (figura a)** oppure l modello **memoria condivisa (figura b)**

Immagine che contiene screenshot

Descrizione generata automaticamente**Spiegazione**  --Nella **figura(a)** si ha uno scambiodi messaggi dal processo A al processo B. Lo scambio avviene in questo modo: Il messaggio di A (casella con la M) viene depositato nel nucleo; dopo di che il nucleo lo spedisce a B. **Vantaggi:** più semplice in quanto la casella postale si trova in una locazione precisa, di conseguenza i processi sanno dove andare a prendere i dati. **Svantaggi:** va bene solo per scambiare piccole quantità dati. -Nella **figura(b)** il nucleo non ha bisogno di intervenire nello scambio, perché i processi A e B hanno una memoria condivisa. **Vantaggi:** posso scambiare grandi quantità di memoria in quanto, essendo la memoria condivisa non si ha bisogno di una parte del nucleo per lo scambio.  **Svantaggi:** I processi si devono conoscere tra loro per sapere dove andare a prendere i dati, dato che la memoria è condivisa, di conseguenza è più complesso.

Programmi di sistema

* I programmi di sistema offrono un ambiente conveniente per lo sviluppo e l’esecuzione dei programmi; in generale si possono classificare in queste categorie:

1. **Gestione dei file**
2. **Informazioni di stato**
3. **Modifica dei file**
4. **Ambienti d’ausilio alla programmazione**
5. **Caricamento ed esecuzione dei programmi**
6. **Comunicazioni**
7. **Programmi d’applicazione**

* Per la maggior parte degli utenti, **l’interfaccia col SO è definita dai programmi di sistema** piuttosto che dalle effettive system call.

Struttura del sistema MS-DOS

* Progettato per fornire la massima funzionalità nel minimo spazio
* Non suddiviso in moduli
* Anche se dotato di una semplice struttura, le sue interfacce e i livelli di funzionalità non sono ben separati.

Immagine che contiene screenshot, segnale

Descrizione generata automaticamente**Spiegazione**

I programmi d’applicazione e i programmi si sistema residenti riescono direttamente a colloquiare con i livelli più bassi by-passando il controllo del SO. Questo era un bene perché lo rendeva veloce, ma un male dato che non c’era un controllo da parte del SO.

Struttura del sistema UNIX (struttura molto modulare)

* Immagine che contiene screenshot

  Descrizione generata automaticamenteLo UNIX è un altro esempio di strutturazione che inizialmente era limitata dalle funzioni dell’architettura sottostante. E’ formato da due parti:

1. **Programmi di sistema**
2. **Nucleo**

* Tutto quello che si trova sotto l’interfaccia delle system call e sopra i dispositivi fisici è il nucleo
* **Fornisce file system, lo scheduling della CPU, la gestione della memoria e altre funzioni riguardanti il SO: in un solo livello sono combinate un’enorme quantità di funzioni.**

Metodo stratificato

* Con il metodo stratificato si suddivide il SO in un certo numero di starti (o livelli), ciascuno costruito sopra gli strati inferiori. **Lo strato più basso (0) è lo strato fisico; quello più alto (strato n) è l’interfaccia d’utente.**
* **Ogni strato si realizza impiegando unicamente le operazioni messe a disposizione dagli strati inferiori.**

**Esempio di uno strato del SO**

**Immagine che contiene screenshot

Descrizione generata automaticamente** Il livello i-esimo fornisce servizi al livello i+1 e mediante system call usa funzionalità del livello i-1

Microkernel (orientamento a micronucleo)

* **Secondo questo orientamento, si progetta il SO rimuovendo da nucleo tutti i componenti non essenziali, realizzandoli come programmi del livello d’utente di sistema.**
* La comunicazione si realizza secondo il modello a scambio di messaggi.
* **Vantaggi:**

-facilità di estensione del SO -più semplice da adattare alle diverse architetture -più affidabile (i servizi si eseguono in gran parte come processi utenti, non come processi del nucleo) -più sicuro

Macchine virtuali

* Il concetto di **macchina virtuale** si sviluppa logicamente dal metodo stratificato. I programmi d’applicazione possono considerare quello che si trova a un livello gerarchico inferiore come se fosse parte della macchina stessa, anche se i programmi di sistema si trovano a un livello successivo
* Una macchina virtuale è un’interfaccia identica all’architettura sottostante
* **Il SO crea l’illusione che un processo disponga della propria CPU con la propria memoria (virtuale)**
* Il calcolatore fisico condivide le risorse in modo da creare una macchina virtuale.
* La partizione del tempo d’uso della CPU si può usare sia per condividere la CPU sia per dare l’illusione che gli utenti dispongono di una propria CPU
* La gestione asincrona delle operazioni di I/O e dell’esecuzione di più processi, unita a un file system, consente di creare lettori di schede e stampanti virtuali
* Un normale terminale si un sistema a partizione del tempo funziona da console d’operatore della macchina virtuale

Immagine che contiene screenshot

Descrizione generata automaticamente**Vantaggi/Svantaggi**

-**l’uso delle macchine virtuali protegge completamente le risorse di sistema poiché ciascuna VM è isolata dalle altre.** Uno svantaggio di questo tipo d’ambiente è che non c’è una condivisione diretta delle risorse. -**un sistema di VM è un perfetto mezzo di ricerca e sviluppo dei SO.** Lo sviluppo avviene sulla VM non sulla macchina fisica, evitando così modifiche che potrebbero causare oscuri errori di programmazione in altri punti. -A una maggiore **complessità della macchina da emulare** corrisponde una maggiore difficoltà di realizzazione di un’accurata VM, e una **maggiore lentezza nell’esecuzione.**