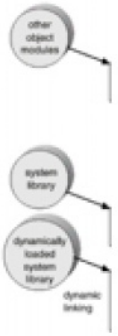
Sistemi Operativi – Capitolo 8

La CPU può accedere solo ai suoi **registri** interni e alla **memoria centrale** (RAM); le istruzioni macchina accettano indirizzi di memoria come argomenti ma **non** indirizzi di un disco. Quindi tutte le istruzioni in esecuzione e i dati devono essere nella RAM per essere eseguite (nella memoria virtuale invece i processi possono essere parzialmente in memoria). Ci sono più processi **concorrenti** in memoria (nella architettura von Neumann), sia utente che di **sistema**, quindi bisogna decidere dove mettere ogni processo (per la protezione) e ci sono 2 registri che determinano ciò:

- **Registro di base** che contiene il più basso indirizzo della memoria fisica (dove parte);
- **Registro di limite** che contiene la dimensione dell'intervallo (dove finisce);

La memoria fuori da questo intervallo **NON** deve essere accessibile, e sarà il sistema operativo a bloccare un accesso al programmma (segnalato). Il S.O. in **modalità di sistema** può accedere alla sua memoria e a quella degli utenti; ovviamente le istruzioni per caricare i registri di base e quelli di limite sono **privilegiate**.

Normalmente un programma utente prima di essere eseguito passa per diversi stati: inizialmente è scritto in un linguaggi ìo ad alto libello poi:



- **Tempo di compilazione**: vengono complilati i vari componenti del codice(cicli, salti) ecc. I risultati del compilatore in genere sono dei **moduli oggetto** passati al linker;
- **Tempo di caricamento**: i linker va ad inserire dell'oggetto altri moduli, o attraverso ul **loader** che va a caricare nell'indirizzo oggetto delle librerie di sistema, il loader crea un codice che già è messo in memoria, e può essere eseguito;
- **Run time**: anche se i processi sono in esecuzione si possono allocare altre librerie **dinamicamente** (dynamic linking).

L'associazione tra **indirizzo logico** e quello **fisico** può essere fatta in almeno una di queste 3 fasi, normalmente si seleziona un processo da una coda di input, viene caricato in memoria, poi quando esce viene deallocata. La maggior parte dei sistemi permette ai **processi utente** di stare in una qualsiasi parte della memoria fisica; un complilatore associa (**Bind**) gli indirizzi simbolici (del programma **sorgente**) a quelli rilocabili e il **loader** collega questi indirizzi simbolici a quelli **assoluti**.

L'associazione di istruzioni e dati ad indirizzi di memoria si può compiere nella fase di:

- **Compilazione** (anche se è molto rara) cioè se si sa già dove il processo andrà in memoria, e quindi si genera direttamente il **codice assoluto** (indirizzi critici); il problema è che se la **locazione iniziale** dovesse cambiare sarebbe necessario **ricompilare** il codice (di nuovo associazione tra indirizzi logici e assoluti);
- **Caricamento** cioè se non si può sapere in compilazione lo spazio, si genera un codice **rilocabile** cioè indirizzi non assoluti ma **relativi**, (es da 0 a n) poi quando si saprà dove verranno caricati gli indirizzi si sommerà **(indirizzo logico+indirizzo rilocazione);** il collegamento finale viene ritardato fino al momento del caricamento e se l' **indirizzo iniziale** cambiasse sarebbe sufficiente caricare solo il **codice utente** ed incorporare il valore modificato;
- **Esecuzione** cioè quando il processo viene passato da un segmento all'altro della memoria finchè il collegamento è **ritardato** (all'esecuzione appunto)