

Schemi di traduzione degli indirizzi

Base e lunghezza

Si mappa lo spazio di indirizzamento logico del processo in una regione contigua di memoria fisica, specificata con indirizzo di base + lunghezza. L'implementazione è semplice (solo due registri per memorizzare base e lunghezza del processo in esecuzione), ma:

- la protezione è limitata a eccezioni per accessi fuori dalla regione indicata, non si possono specificare diritti diversi per aree diverse;
- non è possibile condividere parte della memoria di processi diversi;
- l'allocazione di memoria fisica è contigua (va trovato un segmento di dimensione sufficiente) e preventiva;
- frammentazione esterna, accentuata se si usa politica best fit (crea buchi piccoli inutilizzabili). Deframmentare è costoso: spostare tutta la memoria del processo richiede 80-100 cicli a parola, senza aiuto delle cache.

Segmentazione

Si divide la memoria dei processi in segmenti (testo, dati, stack/heap, SO, eventualmente un segmento per ogni shared library), ciascuno mappato indipendentemente in memoria fisica con base + lunghezza.

Riduce la frammentazione esterna rispetto a un'allocazione singola (le unità di allocazione sono più piccole) e permette di assegnare permessi diversi ai segmenti. Si può fare condivisione di codice (e.g. per fork).

Il sistema mantiene una *tabella dei segmenti* con le associazioni segmento-base-lunghezza-permessi. Gli indirizzi logici sono della forma $\langle \# \text{segmento}, \text{offset} \rangle$, ma non è nota a priori la quantità di bit necessaria per specificarli, per cui devono stare in due registri diversi. La variabilità delle dimensioni dei segmenti in generale porta ad inefficienze nell'implementazione (e a frammentazione esterna).

Accedere ad un indirizzo fuori dai segmenti genera un'eccezione; a seconda delle circostanze questa può provocare l'espansione del segmento (e.g. crescita dello stack), oppure un *segmentation fault*.

Paginazione

Divisione della memoria in *pagine* di dimensione fissa. La tabella delle pagine mantiene associazioni tra pagine logiche e pagine fisiche, protezioni e informazioni di controllo (e.g. bit per segnalare che la pagina deve essere caricata dal disco). Gli indirizzi sono della forma $\langle \# \text{pagina}, \text{offset} \rangle$, e possono essere mantenuti in un singolo registro: con pagine da 4K, 12 di offset e il resto #pagina.

Rende l'allocazione della memoria molto semplice: non c'è bisogno di trovare grandi regioni libere contigue, è sufficiente raccogliere il numero di pagine richiesto (anche in ordine sparso), individuabili rappresentando la memoria fisica come bitmap (libera/occupata).

Oltre ai vantaggi della segmentazione, la paginazione permette caricamento incrementale di codice (o dati) dal disco: si inizia l'esecuzione prima di aver caricato tutto l'eseguibile, se il programma arriva ad un'istruzione contenuta in una pagina non ancora caricata viene generata un'eccezione; il kernel la gestisce sospendendo il programma finché il codice non è disponibile.

La dimensione della tabella delle pagine dipende dalla dimensione della memoria virtuale, per cui su sistemi a 64 bit c'è un overhead significativo visto che questa è molto maggiore di quella fisica (e considerando che ogni processo ha la sua tabella). Si potrebbe risolvere aumentando la dimensione delle pagine, ma questo peggiora la *frammentazione interna* (in particolare se si usa segmentazione con pagine). La soluzione è la paginazione multilivello.