**Page replacement algorithms**

Quando avviene un page fault il sistema operativo deve liberare la memoria scegliendo su quali pagine può eseguire uno swapped-out

**FIFO**

**Immagine che contiene testo

Descrizione generata automaticamente**

**FIFO**  è semplice, ma non è ideale per rimpiazzare le pagine, non utilizza il principio di località

Serve un algoritmo che minimizzi i page faults, non è possibile avere un algo del genere, si deve basare su delle approssimazioni

**NRU(Not recently used)**

Usa 2 bit nella tabella delle pagine

R e M

* R= bit dei riferimenti, indica se la pagina è stata letta o scritta da qualche istruzione vale 1 quando avviene un accesso, il sistema operativo lo mette a 0 la MMU lo mette a 1 se avviene un accesso.

Periodicamente il SO lo resetta a 0

* M=bit di modifica, ha lo scopo di indicare se la pagina è stata modificata. Serve in fase di swapped out per capire se salvare il contenuto su disco.

NRU usa delle classi per operare

Immagine che contiene testo

Descrizione generata automaticamente

**Second chance**

È possibile migliorare FIFO usando il bit R

Associato ad ogni pagina abbiamo R, quando una pagina diventa candidata per la rimozione, se R = 1

Non rimuoviamo la pagina, ma la mettiamo in fondo alla lista e mettiamo R=0

Un miglioramento di second chance è l’algoritmo dell’orologio (clock algorithm) ed implementa code circolari.

Spostare un elemento in fondo non necessita di spostare nessun elemento, ma aggiornare il puntatore della testa (lancietta)

Se il bit R = 1 -> il bit R viene messo a 0 e si va al successivo

Se la pagine ha bit R=0 allora metto la nuova pagina, e l’altra viene salvata su disco

È possibile migliorare la località

**LRU (Least Recently Used)**

Rimpiazza la pagine usata meno recente ciò è possible mettendo un ulteriore etichetta che indica quando è stata fatta l’ultima modifica, il bit R singolo non è sufficiente.

Questo però è costoso in termini di tempo.

LRU può essere messo in Hardware con un contatore

Immagine che contiene tavolo

Descrizione generata automaticamente

Anche qui c’è un costo di accesso in memoria

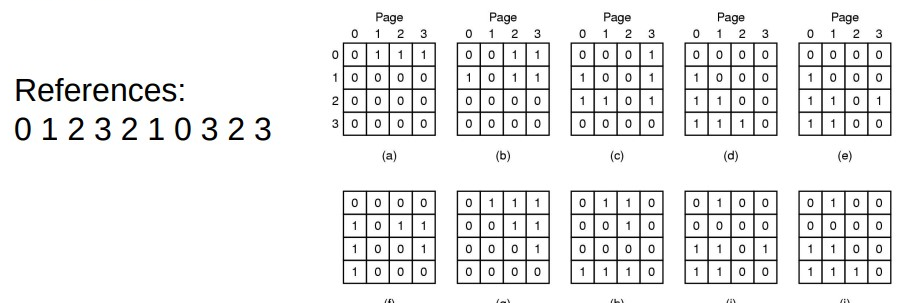
Altra possibilità Hardware usando una matrice anziché un counter

Matrice nxn con n=numero frame

La matrice è inizializzata a 0

Quando la pagina caricata con frame k è referenziata tutti i bit alla riga k sono a 1 e di conseguenza la colonna k è a 0.

La riga con il valore binario più piccolo è quella che è stata modificata meno recentemente.

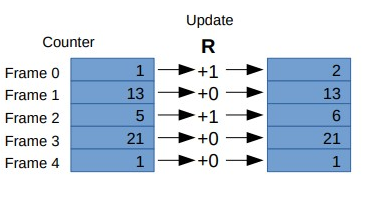


**Soluzione Software**

**NFU (Not Frequently Used)**

Periodicamente guarda il bit R e lo somma a un contatore (ogni pagina ha un contatore) e viene aggiornato periodicamente

Andrà a rimuovere la pagina con il valore più piccolo.



**NFU modificato**

Ha un meccanismo di invecchiamento, il vantaggio guadagnato con il tempo perde valore.

I contatori hanno dimensione fissata

Dopo un periodo shifto a sinistra e sommo

Chi riceve tanti riferimenti, li perderà poiché vengono shiftati.

Immagine che contiene tavolo

Descrizione generata automaticamente

**Working set**

insieme di pagine utilizzate dai k indirizzi di memoria più recenti; w(k,t) è la dimensione del working set all'istante k.

Lavorare con il working set può portare delle ottimizzazioni

**The working set algorithm**

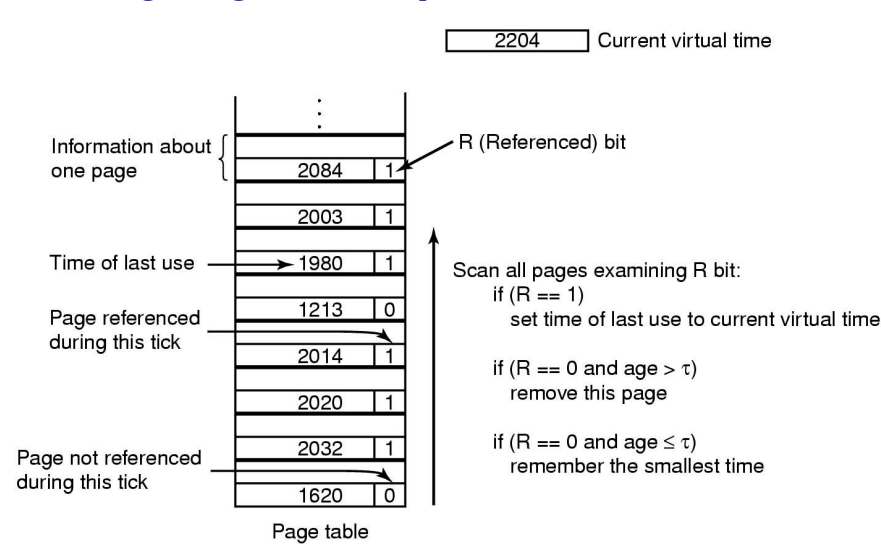
* Manteniamo un timestamp per tenere traccia del tempo di esecuzione virtuale di un processo
* In ogni voce della tabella di pagina è riservato un campo per memorizzare tale timestamp

Quando dobbiamo rimuovere una pagina, cerchiamo nella tabella delle pagine:

* se R=1, copiamo il timestamp corrente nella voce e impostiamo R=0
* se R=0 e il timestamp è più vecchio di una soglia τ la pagina è probabile che non faccia parte del gruppo di lavoro e diventa un candidato per lo sfratto

(il candidato più anziano viene successivamente scelto)

Se tutte le pagine avevano R=1, viene eseguita una scelta casuale



Esiste anche una versione del workingset con l’approccio dell’orologio.

**Se non c’è memoria fisica sufficiente?**

Situazione di trashing

Avviene quando la memoria fisica è piene e ci sono frequenti page faults, il sistema operativo effettua uno swapping aggressivo, coinvolgendo molto la memoria secondaria le prestazioni calano drasticamente.

**Sicurezza**

In memoria scrivono dati provenienti dall’esterno, bisogna impiegare sistemi di protezione della memoria.

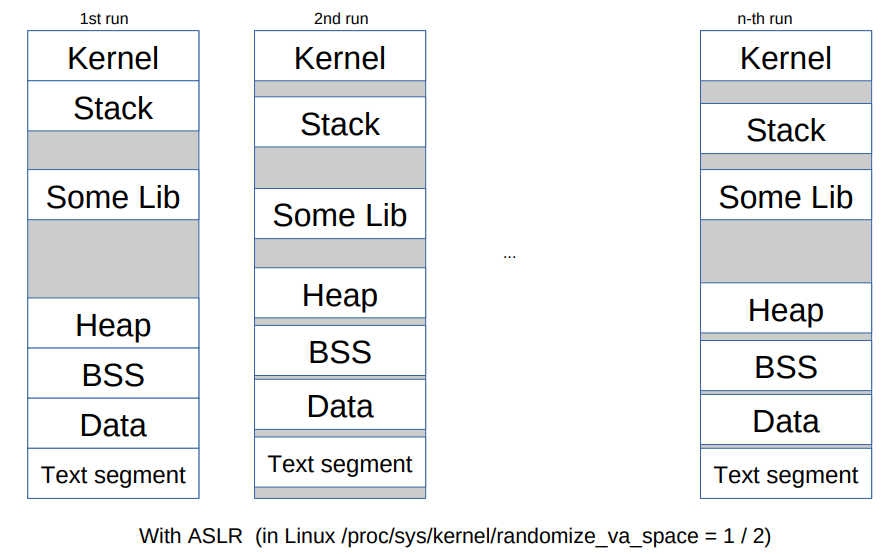
Gli attacchi possono essere effettuati attaccando lo stack tramite funzioni che per esempio non controllano la lunghezza dell’input.

Si può andare a sovrascrivere un return ad esempio eccedendo con la lunghezza di un input.

Mettendo una stringa adhoc è possible saltare in altri punti del software.

**ASLR**

È una strategia che consiste nel randomizzare la memoria nello spazio di indirizzamento.



Aslr sul linux può essere disattivato

**RTOS**bisogna minimizzare l’overhead causato dalla gestione della memoria

Frammentazione di memoria è un problema (è causata dalle continue allocazioni e deallocazioni)

Per evitare la frammentazione si fa un’ottimizzazione tramite pool di blocchi preallocati di dimensione fissa.

Vengono presi quando si alloca e vengono reinseriti nel pool quando si fa il free.