POLITECNICO DI MILANO

Memoria Virtuale

Alessandro A. Nacci
alessandro.nacci@polimi.it

ACSO

2014/2014


START

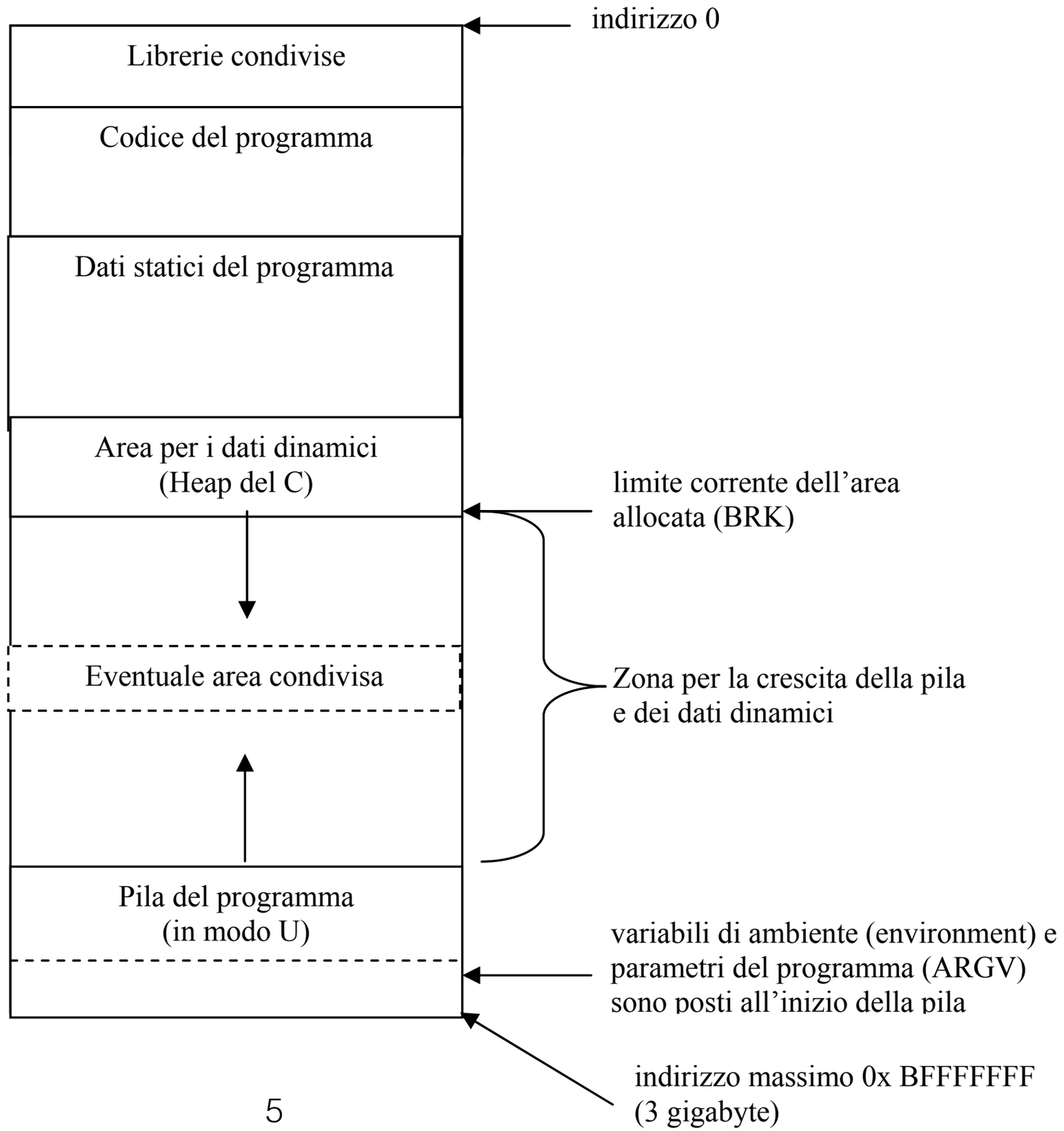
Algoritmo LRU

- Buone prestazioni in media
- Utilizza bit di controllo che riportano le informazioni sugli accessi alle pagine
- La gestione più semplice prevede un bit di accesso nella tabella delle pagine
  - il bit viene posto a 1 quando la pagina viene acceduta e azzerato periodicamente da SO
  - in corrispondenza dell'azzeramento periodico, il SO incrementa una variabile di conteggio interna, per tutte le pagine con bit di accesso a 0
  - viene sostituita la pagina con bit di accesso a 0 e valore di conteggio più alto
- Esistono situazioni particolari in cui l'algoritmo ha prestazioni pessime

Fork()

- L’esecuzione della fork crea un nuovo processo che è l’immagine del padre
- Viene aggiornata la tabella dei processi, ma non viene allocata memoria fisica e i segmenti virtuali vengono condivisi
- All’atto di una scrittura in memoria, da parte di uno dei due processi, la pagina scritta viene effettivamente allocata (le pagine relative a segmenti virtuali contenenti dati vengono inizialmente poste con diritto di accesso in sola lettura)

Struttura Programma Eseguibile



Esercizio Riassuntivo (1)

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 Kbyte, quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte. Si chiede di svolgere i punti seguenti:

a. **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi NPF, Spiazzamento fisico, NPL, Spiazzamento logico

Esercizio Riassuntivo (1)

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 Kbyte, quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte. Si chiede di svolgere i punti seguenti:

a. **Si definisca** la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi NPF, Spiazzamento fisico, NPL, Spiazzamento logico

a) NPF: 3, Spiazzamento fisico: 12, NPL: 3, Spiazzamento logico: 12

Esercizio Riassuntivo (2)

b. Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R, S. I programmi eseguiti da tali processi sono due: X e Y. La dimensione iniziale dei segmenti dei programmi è la seguente:

CX: 8 K DX: 4 K PX: 4 K

CY: 12 K DY: 8 K PY: 4 K

Si inserisca in tabella 1 la struttura in pagine della memoria virtuale (mediante la notazione definita sopra: CX0 CX1 DX0 PX0 … CY0 …).

indir. virtuale	**prog. X**	**prog. Y**
0		
1		
2		
3		
4		
5		
6		
7		

1) memoria logica

Esercizio Riassuntivo (2)

b. Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R, S. I programmi eseguiti da tali processi sono due: X e Y. La dimensione iniziale dei segmenti dei programmi è la seguente:

CX: 8 K DX: 4 K PX: 4 K

CY: 12 K DY: 8 K PY: 4 K

Si inserisca in tabella 1 la struttura in pagine della memoria virtuale (mediante la notazione definita sopra: CX0 CX1 DX0 PX0 … CY0 …).

indir. virtuale	prog. X	prog. Y
0		
1		
2		
3		
4		
5		
6		
7		

1) memoria logica

indir. virtuale	prog. X	prog. Y
0	*CX0*	*CY0*
1	*CX1*	*CY1*
2	*DX0*	*CY2*
3		*DY0*
4		*DY1*
5		
6		
7	*PX0*	*PY0*

Esercizio Riassuntivo (3)

c. A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:
   1. creazione del processo P e lancio del programma Y ("fork" di P ed "exec" di Y)
   2. creazione del processo Q e lancio del programma X ("fork" di Q ed "exec" di X)
   3. accesso a 1 pagina dati e creazione di una nuova pagina di pila da parte di P
   4. accesso a 1 pagina dati da parte di Q
   5. creazione del processo R come figlio di P ("fork" eseguita da P)
   6. creazione di 1 pagina di pila da parte di R

- il lancio di una programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila
- il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti)
- l'indirizzo (esadecimale) dell'istruzione di partenza di X è 14AF
- l'indirizzo (esadecimale) dell'istruzione di partenza di Y è 0231
- il numero di pagine residenti **R** vale **3** (tre)
- viene utilizzato l'algoritmo LRU (ove richiesto prima si dealloca una pagina di processo e poi si procede alla nuova assegnazione)
- le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo, con la sola eccezione seguente: se è residente una sola pagina di codice, quella è certamente stata utilizzata recentemente
- al momento di una fork viene duplicata solamente la pagina di pila caricata più recentemente
- dopo la fork le pagine di codice possono essere condivise tra i processi padre e figlio, se ambedue i processi usano la stessa pagina virtuale

Esercizio Riassuntivo (3) *soluzione*

c) Per facilitare la comprensione del contenuto della tabella 2 (tempo $t_{0)}$, riportato sotto, si forniscono le seguenti spiegazioni, seguendo la numerazione degli eventi:

1. il processo P parte caricando la pagina CP0 (perché il programma Y ha istruzione di partenza in pagina 0) e una pagina di pila PP0
2. successivamente Q carica CQ1 (perché il programma X ha istruzione di partenza in pagina 1) e PQ0
3. P carica DP0, raggiungendo il limite delle pagine residenti (3), e quindi per caricare la pagina PP1 deve eliminare PP0 (vedi regole di utilizzazione indicate nel tema)
4. Q carica DQ0 e raggiunge 3 pagine caricate
5. il nuovo processo R non ha bisogno di caricare CR0, perché esegue lo stesso programma di P e quindi CR0 è uguale a CP0, quindi carica solamente PR1 (che è la copia della pagina PP1, ma conterrà un diverso pid)
6. R carica PR2

indir. fisico	**pagine allocate al tempo t_0**
0	*CP0 (= CR0)*
1	~~*PP0*~~ *PP1*
2	*CQ1*
3	*PQ0*
4	*DP0*
5	*DQ0*
6	*PR1*
7	*PR2*

Esercizio Riassuntivo (4)

d) A un certo istante di tempo $t_1 > t_0$ sono terminati gli eventi seguenti:

7. terminazione del processo P (exit)
8. esecuzione della funzione “exec Y” (lancio di Y) nel processo Q e conseguente trasformazione di Q in processo S (si noti che Q si trasforma in S ma il pid resta lo stesso perché non c’è “fork”)
9. accesso a 2 pagine di dati per il processo S

d) Il contenuto della tabella 3 si spiega nel modo seguente

7. P termina e libera le pagine fisiche 1 e 4 (non la 0, perché CR0 rimane necessaria)
8. la exec Y da parte di Q non alloca il codice, perché CS0 risulta identica a CR0, ma alloca la pagina PS0 nella pagina fisica 1, già libera; vengono inoltre liberate le pagine fisiche 2 e 3
9. la nuova pagina DS0 viene allocata in pagina fisica 2, ma S raggiunge così il livello massimo di pagine residenti e quindi la successiva (DS1) deve essere allocata al posto di PS0

indir. fisico	**pagine allocate al tempo t_1**
0	*CR0 (= CS0)*
1	~~*PS0*~~ *DS1*
2	~~*CQ1*~~ *DS0*
3	~~*PQ0*~~ ---
4	
5	
6	*PR1*
7	*PR2*

Alla prossima lezione

alessandro.nacci@polimi.it