

RIASSUNTO DELLE NOTAZIONI E DELLE REGOLE DELLO *SCHEDULER CFS*

Parametri *SYSCTL* (gestiti da *admin*)

LT: latenza (default 6 ms)

GR: granularità (default 0,75 ms)

WGR: granularità di *wake_up* (default 1 ms)

Altri parametri globali del sistema

tick: ogni 0,01 ms

NOW: istante di tempo corrente

NRT: numero di *task PRONTI* (compreso *CURR*)

Elementi e parametri globali della *runqueue*

CURR: puntatore al *task* corrente

RB: coda dei *task PRONTI* (escluso *CURR*) ordinata secondo i loro *VRT*

LFT: puntatore al *task* con *VRT* minimo in *RB*

IDLE: puntatore al *task idle* (tale *task* non si trova mai nella coda *RB*)

RQL: somma dei pesi *LOAD* di tutti i *task* presenti nella *runqueue* (compreso *CURR*)

VMIN: vedi formula sotto riportata

Parametri di un generico *task t*

t.SUM: tempo reale di exec.

t.VRT: tempo virtuale di exec.

t.LOAD: peso alla creazione *SUM* = 0

t.START: valore di *NOW* al *tick* precedente

t.PREV: valore di *SUM* quando il *task* è diventato *CURR*

Calcolo dei coefficienti di peso (effettivo *LC* e virtuale *VRTC*) di un generico *task t*

$t.LC = t.LOAD / RQL$

$t.VRTC = 1 / t.LOAD$

Calcolo del periodo (finestra) di *scheduling PER* del sistema e calcolo del quanto *Q* di un generico *task t*

$PER = \max(LT, NRT \times GR)$

$t.Q = PER \times t.LC$

Aggiornamento periodico (*task_tick*) del *task* corrente *CURR* a ogni *tick* e verifica della scadenza del suo quanto *Q*

$DELTA = NOW - CURR.START$

$CURR.SUM = CURR.SUM + DELTA$

$CURR.VRT = CURR.VRT + DELTA \times CURR.VRTC$

$CURR.START = NOW$

$VMIN = \max(VMIN, \min(CURR.VRT, LFT.VRT))$

if $(CURR.SUM - CURR.PREV \geq CURR.Q)$ then *resched*

Risveglio (*wake_up* e *wake_up_process*) di un generico *task tw* e verifica della condizione di *preemption*

$tw.VRT = \max(tw.VRT, VMIN - LT / 2)$

if $((tw.schedule_class == \text{diritto di esecuzione maggiore}) \text{ or } (tw.VRT + WGR \times tw.LC < CURR.VRT))$ then *resched*

Creazione (*sys_clone*) di un generico *task tnew* e verifica della condizione di *preemption*

$tnew.VRT = VMIN + tnew.Q \times tnew.VRTC$

if $((tnew.schedule_class == \text{diritto di esecuzione maggiore}) \text{ or } (tnew.VRT + WGR \times tnew.LC < CURR.VRT))$ then *resched*

IPOTESI SULL'ESECUZIONE DEI TASK (PROCESSI) PER GLI ESERCIZI SULLE COMMUTAZIONI DI STATO

Valgono le seguenti ipotesi di *scheduling* semplificato e coerente con quanto approfondito nel capitolo relativo:

- a ogni evento di creazione di un *task*, il *task* che ha creato (padre / default) prosegue l'esecuzione, se ciò è possibile
- in caso di attivazione dello *scheduler*:
 - se l'insieme dei *task* in stato di *PRONTO* è stato modificato, il *task* che ha subito la transizione *ATTESA* → *PRONTO* va direttamente in esecuzione
 - altrimenti va in esecuzione il *task* che è tra quelli in stato di *PRONTO* da più tempo, *senza considerare* il *task idle* a meno che non sia l'unico *task* pronto

Quando un *task* viene messo in stato di *ATTESA*, tra parentesi va indicato il nome della chiamata di sistema / libreria che ha provocato la transizione di stato.

Nella simulazione temporale dell'esecuzione del codice:

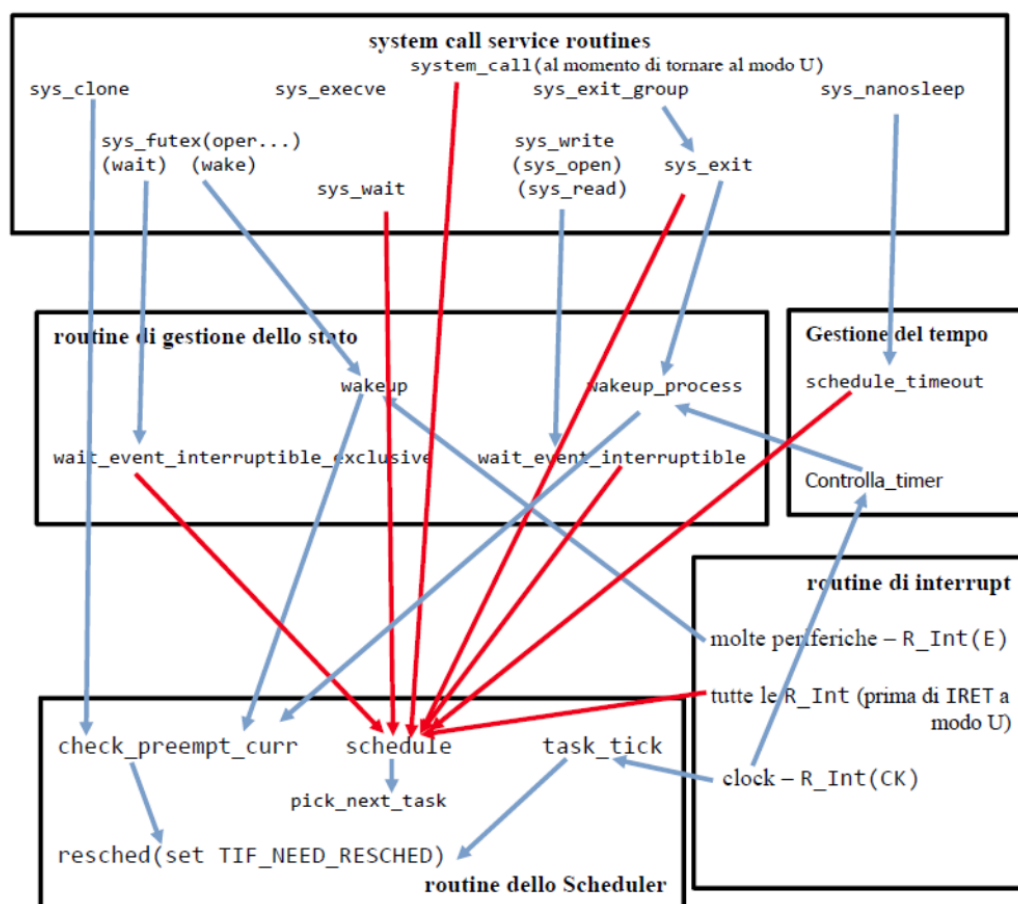
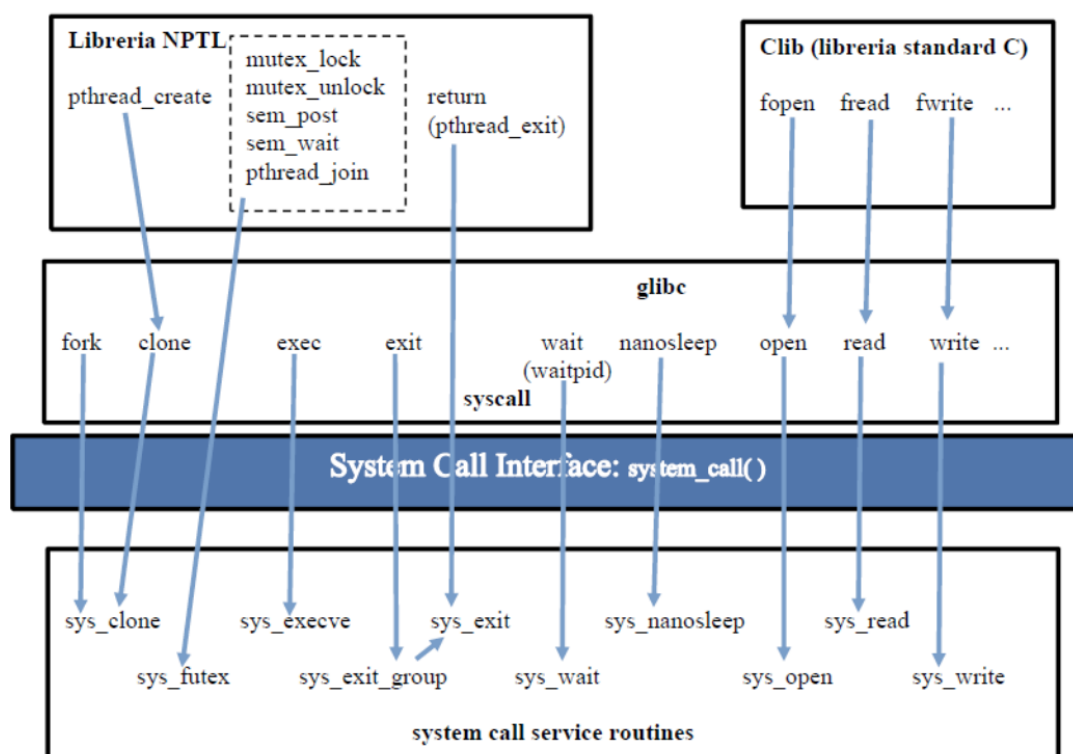
- sono da considerare solo le funzioni marcate in **grassetto**
- l'esecuzione della funzione *exec* non sospende il processo
- l'esecuzione delle funzioni *open*, *read*, *write* e *close* implica sempre l'accesso a disco

Libreria Clib – Libreria standard **C ANSI**

Libreria NPTL – **Native POSIX Thread Library**

Libreria glibc – **GNU C Library** (invocano `syscall` per passare a modo S ed entrano nel SO attivando `System_Call`)

System Call Service Routine (la funzione `System_Call` è il punto di ingresso unico nel SO per tutti i servizi)



IPOTESI E CONVENZIONI PER SIMULARE IL FUNZIONAMENTO DELLA MEMORIA

Quali e quante pagine considerare, e quali righe di *TLB* sono disponibili

La simulazione riguarda solo le pagine dei processi in modo *U*. Il numero totale di pagine fisiche disponibili (in modo *U*) non varia mai durante la simulazione. Si suppone che il *TLB* abbia sempre una riga libera disponibile per una nuova pagina.

- L'inserimento nel *TLB*, nella memoria e nello *swap file* va fatto a partire dalla prima posizione libera.
- Quando un elemento del *TLB*, della memoria fisica o dello *swap file* risulta libero, lasciarlo bianco.

Gestione del *DIRTY bit* del *TLB*

Quando una riga del *TLB* si libera, l'indicazione *D* va riportata nelle *PTE* che la referenziano e nel descrittore di pagina fisica.

Rappresentazione della Tabella delle Pagine (*TP*) di un processo

Quando richiesta, deve essere rappresentata come un elenco di *PTE* contenenti le seguenti informazioni / valori:

- **NPV** (cioè il Numero di Pagina Virtuale)
- Numero di pagina fisica, oppure - se la pagina è in *swap file* - l'indicazione **Sn**, dove *n* è il numero di *page slot* nello *swap file*
- **D** se la pagina è stata scritta
- Protezione (**R** o **W**)

NPV da assegnare al registro *PC* e al registro *SP*

Sono rispettivamente gli NPV dell'ultima pagina di codice acceduta e dell'ultima pagina di pila acceduta.

Ordinamento delle *VMA* di pila dei *thread* creati

Sono da presentare in ordine inverso di creazione (dimensione 2 pagine, con una pagina di interposizione).

Convenzioni di gestione delle *LRU LIST*

- le nuove pagine caricate sono poste in testa alla *Active*, con bit *ref* = 1
- le pagine del processo figlio creato da *fork* sono poste in testa alla *Active* o alla *Inactive* nello stesso ordine di quelle del padre e con lo stesso valore del bit *ref*
- le eventuali pagine condivise con quella caricata a seguito di *swap_in* sono poste in coda alla *Inactive* con *ref* = 0
- se necessario, le eventuali pagine da scaricare vanno selezionate a partire dalla coda della *Inactive*, tenendo conto del vincolo imposto dalle pagine condivise

Allocazione di pagine fisiche

Siano *FREE* le pagine libere della memoria fisica, *MAX_FREE* e *MIN_FREE* i due parametri di sistema, e *REQUIRED* le pagine da allocare. Se $FREE - REQUIRED < MIN_FREE$, viene attivato l'algoritmo *PFRA*, che riporta *FREE* a $MAX_FREE + REQUIRED$, e quindi vengono allocate le pagine.

N.B.: nelle simulazioni si ipotizza sempre che le pagine da allocare vengano richieste una alla volta (*REQUIRED* è sempre 1).

Comportamento del daemon *kswapd*

All'attivazione periodica, il daemon *kswapd* effettua in ordine queste due operazioni:

- gestisce le *LRU LIST* tramite la funzione *Controlla_liste*
- se $FREE < MAX_FREE$, esegue l'algoritmo *PFRA* che riporta *FREE* a *MAX_FREE*

Comportamento dell'algoritmo *PFRA*

PFRA scarica un numero adeguato di pagine, agendo prima sulle pagine di *page cache* che non sono mappate su pagine di un processo e poi sulle pagine presenti nella *LRU LIST Inactive*.

Evento composto: *Read (lista pagine lette) – Write (lista pagine scritte) – N kswapd*

Ripete *N* volte le operazioni di lettura / scrittura / *kswapd* ed esegue un'ultima volta le sole operazioni di lettura / scrittura.

Nota bene: se *N* = 0 vengono eseguite una sola volta le operazioni di lettura / scrittura; se la lista di operazioni è vuota, viene eseguito *N* volte *kswapd*.

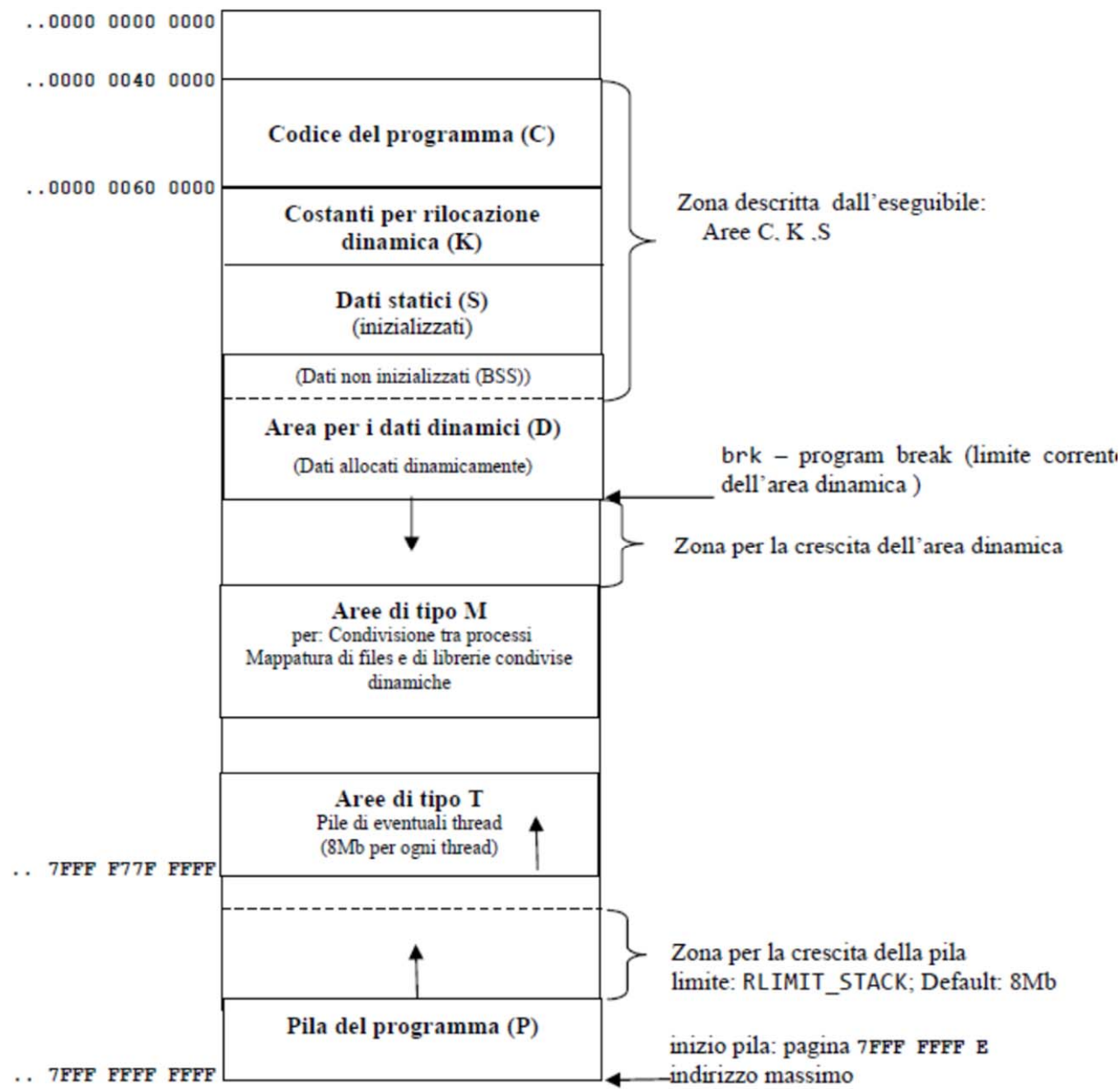
Evento: *mmap (ind, dim, R/W, P/S, M/A, "nomefile", offset)*

Parametri - **ind**: indirizzo iniziale della *VMA*, **dim**: dimensione in pagine della *VMA*, **R/W**: modalità di accesso consentito, **P/S**: *private/shared*, **M/A**: mappata/anonima, **"nomefile"**: nome del file mappato, **offset**: dall'inizio del file della prima pagina mappata. Se l'area è anonima i valori di default degli ultimi due parametri sono -1 e 0, rispettivamente

Evento: *exec (nC, nK, nS, nD, pagina virtuale della prima istruzione del codice, "nomefile")*

Parametri - **nC**, **nK**, **nS**, **nD**: numero di pagine iniziali delle corrispondenti aree, **"nomefile"**: nome del file eseguibile. Viene automaticamente creata un'area di pila (**P**) di 3 pagine

MODELLO DI MEMORIA VIRTUALE DEL TASK IN SPAZIO UTENTE (USERSPACE)



SPAZI DI INDIRIZZAMENTO DI UTENTE E DI SISTEMA

Indirizzi virtuali in modo U: da 0000 0000 0000 0000 a 0000 7FFF FFFF FFFF

Indirizzi virtuali in modo S: da FFFF 8000 0000 0000 a FFFF FFFF FFFF FFFF

spazio libero per brutta copia o continuazione

spazio libero per brutta copia o continuazione

spazio libero per brutta copia o continuazione

spazio libero per brutta copia o continuazione