

2 – Processi e Thread

Sommario

Processi

modello
operazioni: creazione, chiusura
gerarchie
stati, ciclo di vita
transizioni di stato
descrittore di processo Process Control Block (PCB)
sospensione, ripresa, cambio di contesto
Interrupt
comunicazione tra processi: segnali e messaggi

Thread

modello e uso

Scheduling

Obbiettivi
Scheduling di processi: algoritmi
Vari tipi di sistemi
Scheduling di thread

S. Balsamo – Università Ca' Foscari Venezia – SO2.39

39

Obbiettivi

- Motivazioni per creare i **thread**
- le somiglianze e le differenze tra i **processi e thread**
- i diversi **livelli di supporto** per i thread
- il **ciclo di vita** di un thread
- segnalazione e cancellazione di thread
- le basi di thread Linux e Windows

S. Balsamo – Università Ca' Foscari Venezia – SO2.40

40

Motivazioni per l'uso dei thread

Fattori che hanno influito sullo sviluppo dei threads

–Progetto Software

- Modularità → parallelismo
- Esprimere in modo più naturale attività intrinsecamente **parallele**
- **Concorrenza**
- Alcune attività si sospendono temporaneamente

–Prestazioni

- Si adatta meglio ai sistemi multiprocessore
- Il thread è **più leggero** del processo, più veloce da gestire
- Migliore sfruttamento se i thread sono I/O bound (non CPU bound)

–Cooperazione

- **condivisione** dello **spazio degli indirizzi** e dei **dati** → riduzione dell'overhead dovuto alla IPC

S. Balsamo – Università Ca' Foscari Venezia – SO2.41

41

Definizione di Thread

- Thread
 - Processo **lightweight** (LWP)
- **Threads** → flusso di **istruzioni** o flusso di **controllo**
- condividono con il proprio processo lo **spazio di indirizzamento, file aperti** e altre informazioni globali
- Registri, stack, maschere dei segnali e altri dati specifici del thread (TSD *Thread Specific Data*) sono locali per ogni thread
- I threads possono essere gestiti
 - dal S.O.
 - o da un'applicazione utente

Esempi:

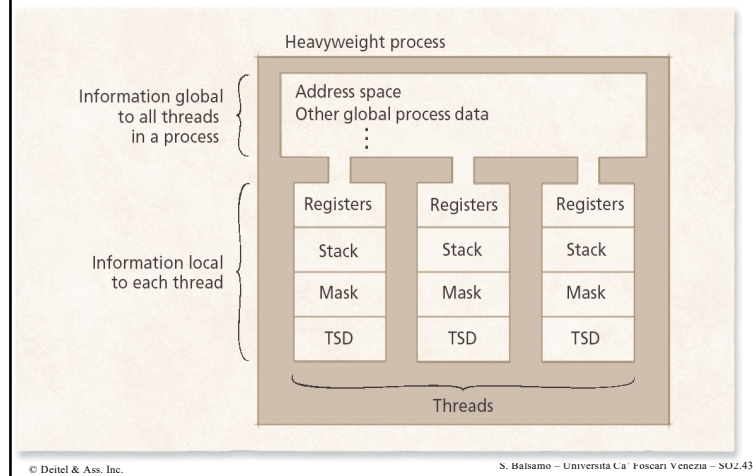
Win32 threads – sistemi Windows a 32 bit
C-threads – libreria microkernel Mach, anche su Solaris, Windows NT
Pthreads – specifica POSIX portabile su vari S.O.

S. Balsamo – Università Ca' Foscari Venezia – SO2.42

42

Definizione di Thread

Relazione fra Thread e Processo



43

Motivazioni ai thread

- Ogni **thread** si sposta fra una serie di **stati discreti**
- Thread e processi hanno molte operazioni in comune (es.: creazione, terminazione, ripresa e sospensione)
- La **creazione** di thread non richiede al S.O. di inizializzare le risorse condivise tra i processi padri e i suoi thread
 - Riduce l'overhead** di **creazione** e **terminazione** dei thread rispetto alla creazione e terminazione di un processo

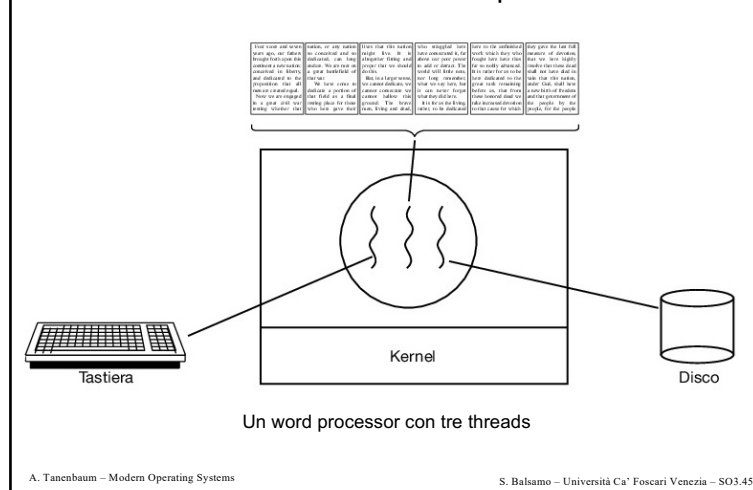
Esempi di utilizzo di thread:

Server (es. web server)

Word processors

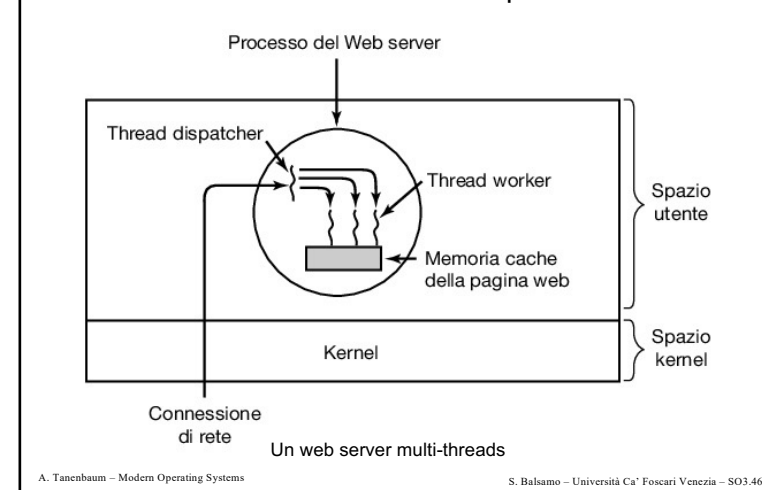
44

Uso di Thread - Esempio 1



45

Uso di Thread - Esempio 2



46

Uso di Thread - Esempio 2

```

while (TRUE) {
    get next request(&buf);
    handoff work(&buf);
}

while (TRUE) {
    wait for work(&buf);
    look for page in cache(&buf, &page);
    if (page not in cache(&page))
        read page from disk(&buf, &page);
    return page(&page);
}

```

(a) Thread dispatcher (b) Thread worker

Un web server **multi-thread**: schema del codice

Multiprogrammazione
Incremento delle prestazioni

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.47

47

Differenze fra thread e processi

Un **processo** contiene e **raggruppa risorse** in relazione fra loro
 spazio di indirizzamento con testo/dati/pila
 altre risorse
 file
 processi figli
 stato e segnali
 account
 ...

Un **thread** è esecuzione di un programma
 è **schedulato** per essere eseguito sul processore
 Program Counter
 registri per le variabili
 stack per chiamate di procedura
condivide lo spazio di indirizzamento e altre risorse con gli altri
 thread dello stesso processo

Multithreading
 in uniprocessori: parallelismo virtuale dei thread

S. Balsamo – Università Ca' Foscari Venezia – SO3.48

48

Modello Thread

Elementi per processo	Elementi per thread
Spazio degli indirizzi	Contatore di programma
Variabili globali	Registri
File aperti	Stack
Processi figli	Stato
Allarmi in sospeso	
Segnali e gestori dei segnali	
Informazioni relative agli account	

Elementi **condivisi** da tutti i thread in un processo - Elementi **privati** ad ogni thread

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.49

49

Stati di un thread: ciclo di vita

- Stati del thread
analoghi agli stati del processo

- Più in dettaglio possiamo distinguere gli stati

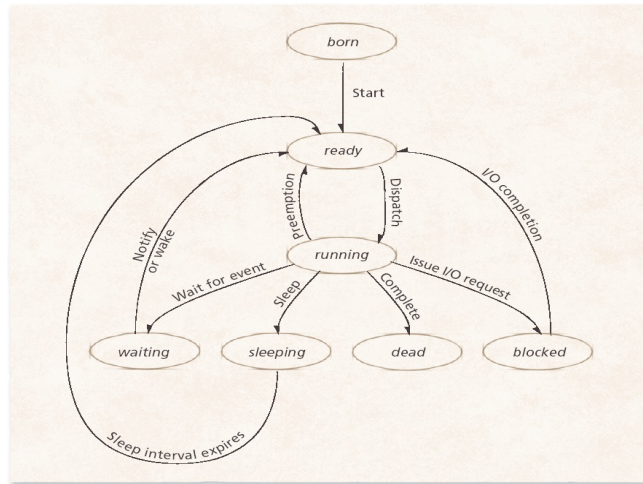
- **Born**
- **Ready** (stato **pronto**)
- **Running** (stato **in esecuzione**)
- **Dead**
- **Blocked** (stato **bloccato da un evento di I/O**)
- **Waiting** (stato **bloccato da un evento di un altro thread**)
- **Sleeping** (stato **bloccato per un tempo prefissato**)
 - L'intervallo di *sleep* specifica per quanto tempo un thread rimarrà in tale stato



S. Balsamo – Università Ca' Foscari Venezia – SO4.50

50

Stati del Thread: ciclo di vita



© Deitel & Ass, Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO4.51

51

Operazioni sui Thread

- Threads e processi hanno delle operazioni in comune
 - Creazione
 - Exit (*terminazione*)
 - Sospensione
 - Recupero (*resume*)
 - Sleep
 - Risveglio

S. Balsamo – Università Ca' Foscari Venezia – SO4.52

52

Operazioni sui Thread

- Le operazioni sui Thread **non** corrispondono **esattamente** alle operazioni sui processi
 - Cancellazione**
 - Indica che un thread dovrebbe essere terminato, ma non è garantito che lo sarà
 - I thread possono disabilitare o mascherare i segnali di cancellazione (eccetto la *abort*)
 - Join**
 - Se esiste un **thread primario** quando termina allora termina anche il processo
 - Il thread primario può dover aspettare che tutti gli altri thread siano terminati. In tal caso ha fatto una *join* con i thread creati da lui
 - Se un thread A fa una *join* con il thread B, allora A non va in esecuzione finché B non ha terminato

S. Balsamo – Università Ca' Foscari Venezia – SO4.53

53

Standard POSIX

- Standard IEEE (IEEE 1003.1.c)
- Obiettivo: **portabilità**
- Package** di thread **Pthread**
- Usato dalla prevalenza dei sistemi Unix
- Include oltre 60 chiamate di sistema
- I thread hanno
 - un identificatore
 - un insieme di registri (PC,...)
 - un insieme di attributi (parametri di scheduling, di stack,...)

S. Balsamo – Università Ca' Foscari Venezia – SO3.54

54

Standard POSIX

Chiamata	Descrizione
Pthread_create	Crea un nuovo thread
Pthread_exit	Termina il thread chiamante
Pthread_join	Attende che un thread specifico esca
Pthread_yield	Rilascia la CPU perché venga eseguito un altro thread
Pthread_attr_init	Crea e inizializza la struttura attributi di un thread
Pthread_attr_destroy	Rimuove la struttura attributi di un thread

Alcune chiamate di Pthread

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.55

55

Modelli di thread

- Tre modelli di threading
 - Threads **a livello di utente**
 - Threads **a livello di kernel**
 - Combinazione** (ibridi) di threads a livello di utente e a livello kernel

S. Balsamo – Università Ca' Foscari Venezia – SO4.56

56

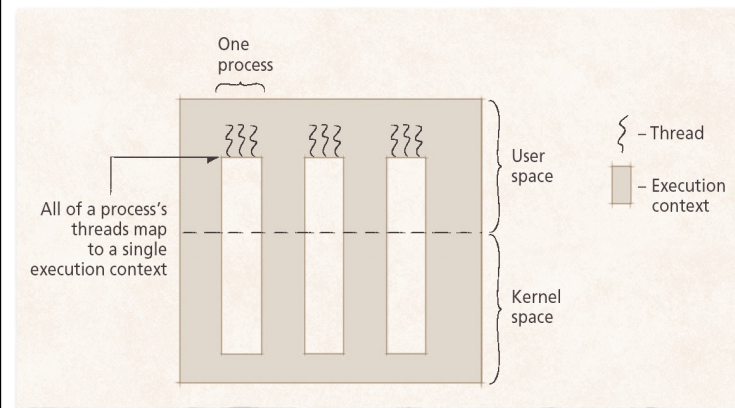
Thread a livello utente

- Threads a livello utente eseguono operazioni di threading nello **spazio utente**
 - Threads sono **creati da librerie a tempo di esecuzione** che non possono eseguire istruzioni privilegiate o accedere direttamente al nucleo
- Implementazione** dei thread a livello utente
 - Mapping di thread **molti-a-uno**
 - S.O. associa tutti i threads di un processo multithread ad un **unico contesto di esecuzione**
- Vantaggi**
 - le librerie di livello utente possono **schedulare** thread per ottimizzare le **prestazioni**
 - La **sincronizzazione** avviene al di fuori del kernel, evitando il cambio di contesto
 - Permette i thread (da utente) dove il s.o. non supporta i thread (interni)
 - Portabilità**
- Svantaggi**
 - Il nucleo vede un processo multithread come **un singolo thread** di controllo
 - Può portare a prestazioni non ottimali se un thread ha problemi di I/O
 - Non può essere schedato su più processori in una sola volta

S. Balsamo – Università Ca' Foscari Venezia – SO4.57

57

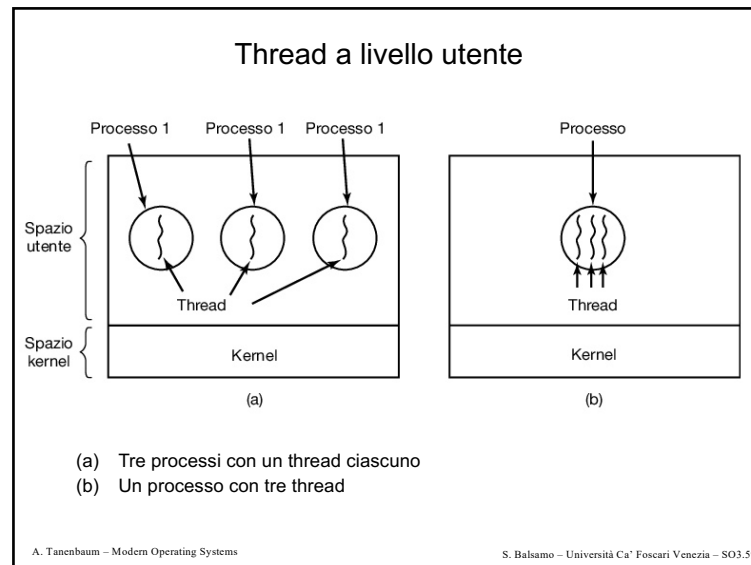
Thread a livello utente



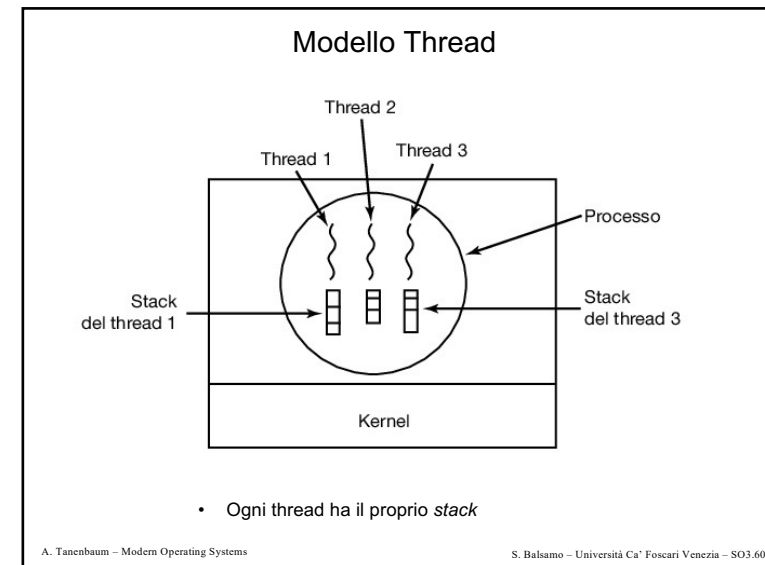
© Deitel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO4.58

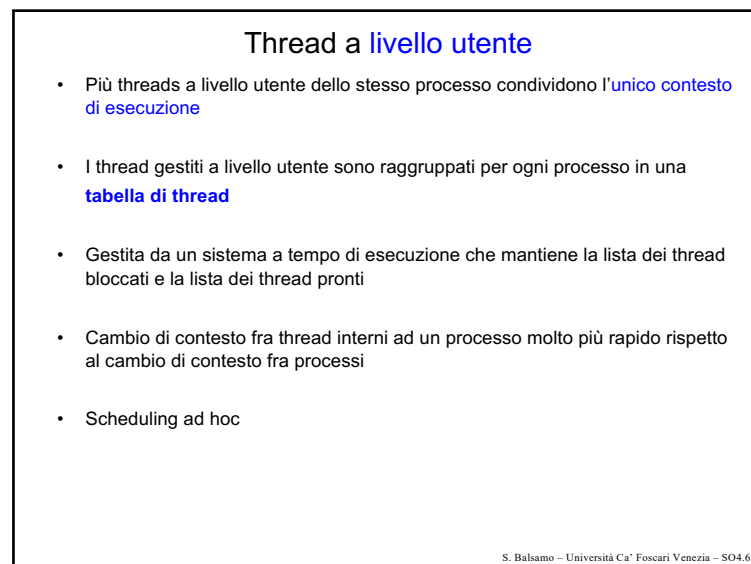
58



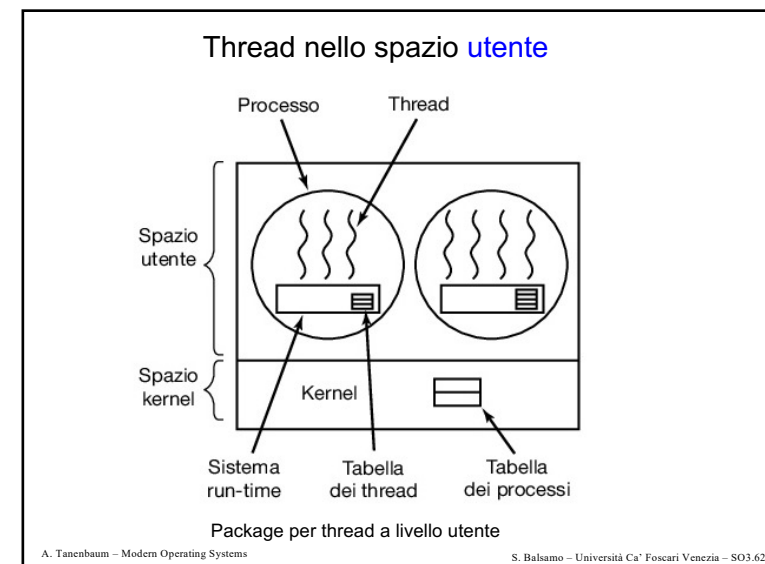
59



60



61



62

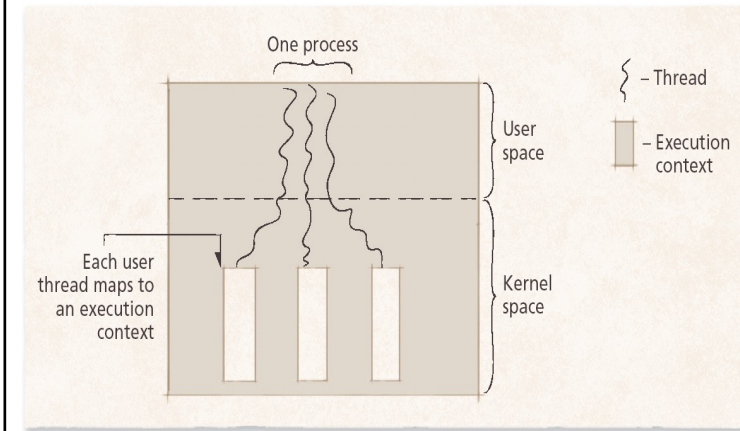
Thread nello spazio kernel

- I thread a livello nucleo cercano di superare i limiti del thread a livello utente **mappando ogni thread** al proprio **contesto di esecuzione**
 - I thread a livello di kernel forniscono un mapping di thread **uno-a-uno**
- Implementazione dei thread a livello utente
 - **Vantaggi:** Aumento della **scalabilità**, **interattività**, e **throughput**
 - **Svantaggi:** **overhead** dovuti al cambio di contesto e **ridotta portabilità** dovuto alle API specifici per S.O.
- I thread a livello nucleo non sono sempre la soluzione ottima per le applicazioni multithread

S. Balsamo – Università Ca' Foscari Venezia – SO4.63

63

Thread a livello kernel



© Deitel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO4.64

64

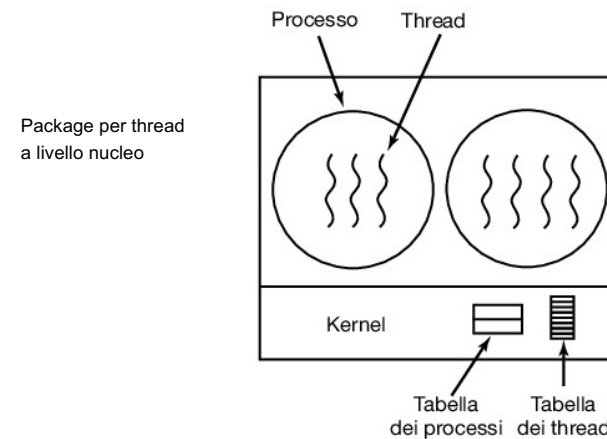
Thread nello spazio kernel

- Non occorre una tabella di thread e un sistema di esecuzione real-time per in ogni processi
- Il nucleo ha la **tabella di thread** con i registri, stato e informazioni di tutti i thread
- Le operazioni sui thread avvengono con **chiamate** al nucleo costo maggiore del cambiamento di contesto possibile 'riciclo' di thread (strutture)
- Lo scheduler è a livello nucleo e può confrontare thread di processi diversi un thread bloccato può non bloccare tutto il processo
- Il nucleo ha anche la **tabella di processi**

S. Balsamo – Università Ca' Foscari Venezia – SO4.65

65

Thread a livello kernel



A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.66

66

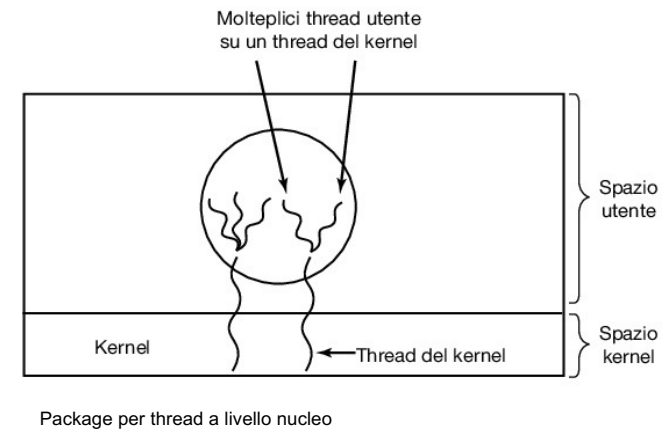
Threads combinati a livello utente e nucleo

- La combinazione di implementazione di thread di livello utente e livello nucleo
 - Mapping thread **multi-a-molti** (*m-to-n* thread mapping)
 - Numero di thread di livello utenti e livello kernel non deve essere uguale
 - Può ridurre l'overhead rispetto al mapping di thread uno-a-uno implementando il **thread pooling**
- Worker threads**
 - threads di livello nucleo **persistenti** che occupano il pool di thread
 - migliora le prestazioni in ambienti in cui i threads sono spesso creati e distrutti
 - ogni nuovo thread viene eseguito da un thread worker
- Attivazione dello **Scheduler**
 - Tecnica che permette alla libreria di livello utente di programmare i suoi thread
 - Si attua quando il S.O. **chiama una libreria di threading a livello utente** che determina se uno qualsiasi dei suoi thread devono essere rischedulati

S. Balsamo – Università Ca' Foscari Venezia – SO4.67

67

Thread ibridi a livello utente e nucleo



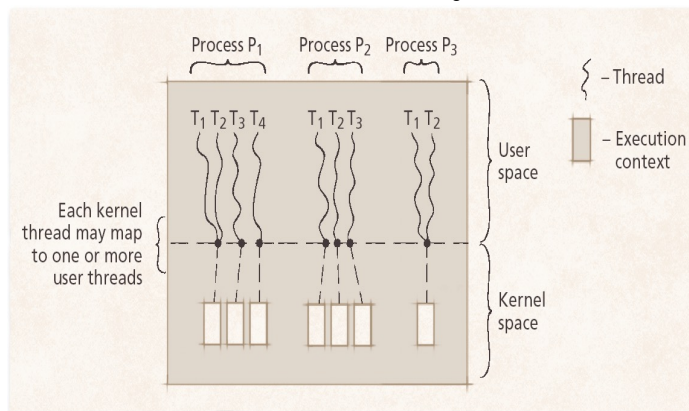
A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.68

68

Thread ibridi a livello utente e nucleo

Modello ibrido di threading



© Deitel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO4.69

69

Attivazione dello Scheduler

Obiettivo

- mimare le **funzionalità** dei threads di livello nucleo
- migliorare le **prestazioni** dei thread del livello utente

Evitare transizioni non necessarie utente/kernel

Il nucleo assegna **processori virtuali** ad ogni processo

- Permette al sistema di allocare a tempo di esecuzione thread ai processori (a livello utente)
- Segnalazioni dal nucleo per possibili situazioni di thread bloccati (upcall) al sistema run-time che può attivare lo scheduler a livello utente

Si può usare su multiprocessori (con processori reali)

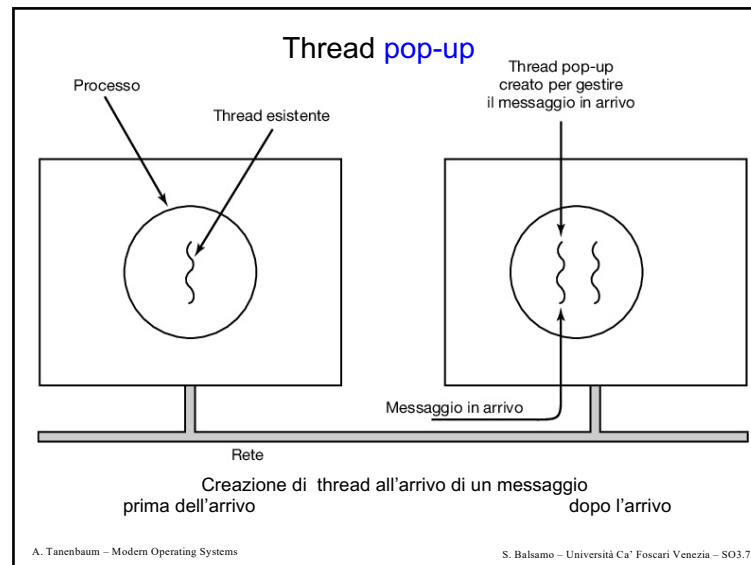
Problema

- Uso intensivo del nucleo (basso livello)
- che chiama procedure nello spazio utente (alto livello)

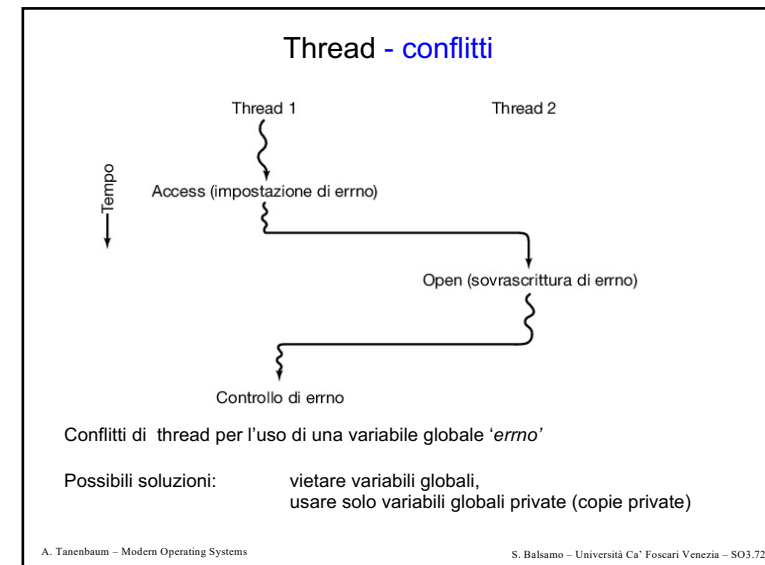
A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO3.70

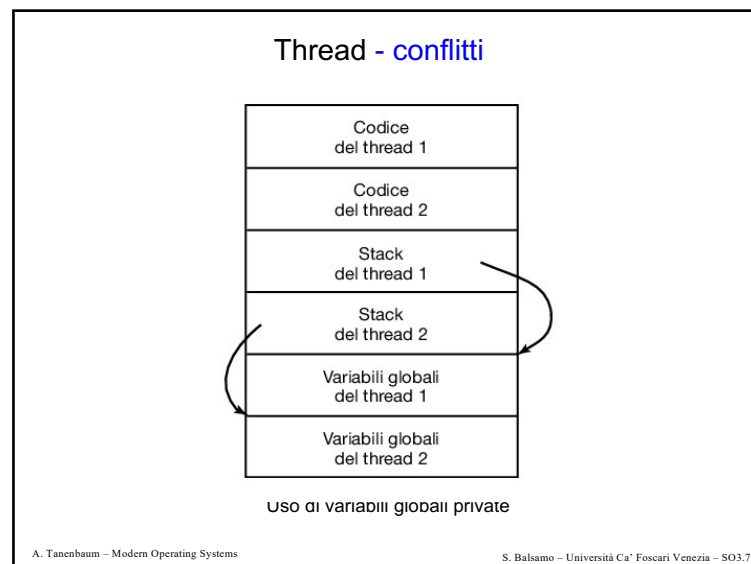
70



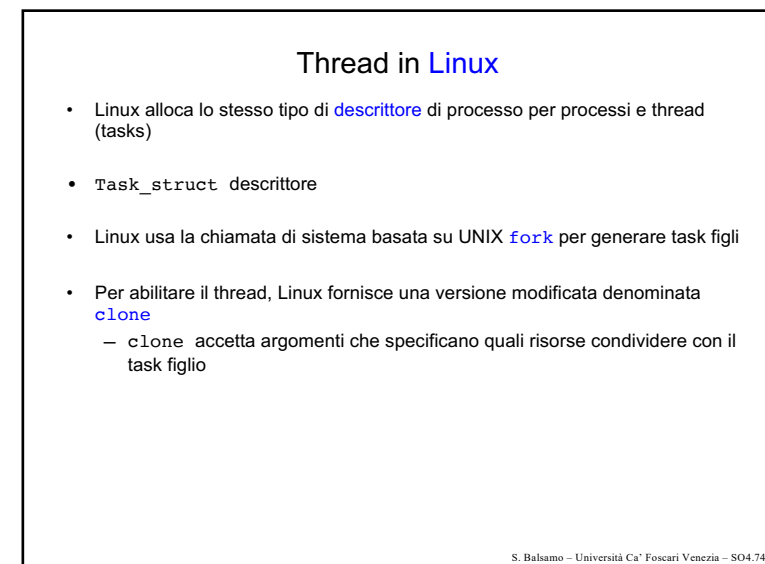
71



72



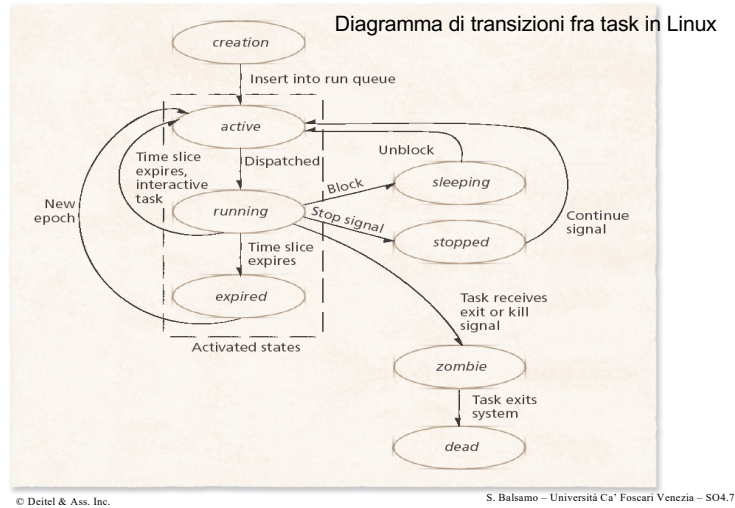
73



74

Thread in Linux

Diagramma di transizioni fra task in Linux



75

Threads Windows

- I processi contengono i programmi, gli handle (riferimenti ad oggetti) e i Thread con cui condividono risorse
 - Thread: unità corrente di esecuzione assegnata ad un processore
 - Esegue un pezzo di codice del processo nel contesto del processo, utilizzando le risorse del processo
 - Il contesto di esecuzione contiene
 - Runtime stack
 - Stato dei registri della macchina
 - Molti attributi
 - unità reale di esecuzione inviato ad un processore
- PEB Process Environment Block
- TEB Thread Environment Block

S. Balsamo - Università Ca' Foscari Venezia - SO4.76

76

Threads Windows

- Windows può raggruppare i processi in *job* per limitare e gestire l'uso di risorse per tutti i thread del processo del job. I *job* contengono processi
- threads possono creare e *fiber* allocando strutture dati e uno stack a livello utente
 - *Fiber* viene eseguito nel contesto del thread che lo crea, invece che lo scheduler
 - I thread possono essere convertiti in *fiber*, le *fiber* possono essere create indipendentemente dai thread
 - Vantaggio per cambio di contesto solo a livello utente
- La relazione *thread fiber* è molti a molti, ma di regola un thread è associabile a un insieme di *fiber*
 - Non sempre usate
- Windows fornisce ad ogni processo un *pool di thread* che si compongono di un numero di *thread worker*, che sono thread di livello *kernel* che eseguono funzioni previste dal thread utente

S. Balsamo - Università Ca' Foscari Venezia - SO4.77

77

Threads Windows

- *Thread pool* funzionalità di Win32
 - Coda di task da eseguire
 - I thread del pool appena liberi prendono un task da eseguire dalla coda
- Evita la creazione/distruzione continua di thread
- I thread possono bloccarsi in attesa di eventi e in quella fase non possono essere riassegnati ad una altro task da eseguire
- Esempio: in applicazioni di tipo cliente-servente

S. Balsamo - Università Ca' Foscari Venezia - SO4.78

78

