PROCESS DESCRIPTION AND CONTROL

Diapositiva 3
Richieste che il sistema operativo deve soddisfare

Diapositiva 4
i processi in user mode per esempio non possono fare accesso diretto all'hardware. se devono farlo effettuano delle chiamate al kernel dette System Call, simili alle chiamate di procedura con la differenza che qui cambia anche la modalità di esecuzione del processore da user a kernel.

Diapositiva 5
Motivi per la creazione di processi:
1-Banalmente il SO passa al processo successivo di una sequenza di processi da eseguire
2-Un utente si logga
3-Per l'esecuzione di un certo processo il SO crea un altro processo per conto di una qualche applicazione.
4-Un processo ne crea altri per motivi di parallelismo, fork ecc.

Diapositiva 6
la chiamata di sistema di normal completion è detta EXIT
negli altri casi in genere è una KILL

Diapositiva 7
ATTENZIONE
Time Limit Excedeed vuol dire che è scaduto il tempo assegnato al processo mentre Time Overrun vuol dire che è scaduto il tempo massimo di attesa da parte del processo per un altro evento.

Diapositiva 11
avviene lo scheduling per decidere quale processo far partre e poi il dispatching. in questo esempio per evitare la monopolizzazione del processore da parte dei processi il SO ha deciso che ognuno di questi possa eseguire al massimo 6 operazioni.

Diapositiva 12
In una architettura particolarmente semplice possiamo immaginare che i processi possano trovarsi solamente o nello stato running o nello stato not running. il SO crea un programma e lo pone come not running in una qualche coda di attesa; il processo in esecuzione ad un certo punto passerà nello stato not running e quello creato nello stato running.
in generale possiamo immaginare questa coda come una lista di puntatori a processi, oppure una lista di blocchi di dati. L'allocatore si occupa di selezionare il processi dalla coda e di gestirli secondo una qualche politica

Diapositiva 13
La semplice coda vista in precedenza non funziona in tutti i casi. se un processo ha lasciato il processore a è tornato in coda perché sta aspettando un evento di I/O può capitare che torni in testa alla coda quando questo evento non si è ancora verificato è quindi non può comunque eseguirsi. Bisogna quindi distinguere oltre a Running e Not-Running anche i casi Ready e Blocked. Generiamo quindi un modello a 5 stati come quello in figura.
Quando un processo è in esecuzione se questa si interrompe per motivi di timeout e quindi il processo rimane nello stato Ready allora viene reimmesso nella coda dei Ready in attesa di esecuzione. Se invece il processo si è interrotto perchè è in attesa di un qualche evento di I/O allora viene inserito in una coda di Blocked e passerà nei Ready quando l'evento si verificherà.

Diapositiva 15
Quando si verifica un evento di I/O il dispatcher cerca nella coda dei bloccati il processo che aspettava questo evento e lo sposta nella coda dei Ready.

Diapositiva 16
In un grande sistema operativo ha molto più senso immaginare che le code di attesa dell' I/O siano più di una, ognuna relativa ad un certo evento, in modo che al verificarsi di tale evento tutti i processi che ne erano in attesa passino nella coda dei Ready contemporaneamente, senza che il dispatcher debba mettersi a scandire la coda dei bloccati.

PROCESS DESCRIPTION AND CONTROL

Diapositiva 17
L'I/O è molto più lento del processore quindi potrebbe succedere che molti processi siano nella coda dei bloccati e nessun altro possa entrare in esecuzione perchè la memoria è piena. La soluzione è considerare un nuovo stato SUSPEND per cui quando un processo passa nello stato suspend viene swappato sul disco e la memoria viene liberata per accogliere un nuovo processo che possa eseguirsi.

Diapositiva 21
heap, contiene dati che alla fine del processo devono restare, mentre i dati contenuti nello stack spariscono alla disattivazione del processo

Diapositiva 24
PCB contiene informazioni sull'identificazione del processo, sullo stato del processo e altre utili per il controllo del processo.
IDENTIFICAZIONE:
-identificatori
STATO
-registri visibili all'utente
-registri di controllo e stato
-puntatori allo stack
CONTROLLO
-schedulazione e informazioni di stato
-strutturazione dei dati
-comunicazione fra i processi
-privilegi dei processi
-gestione della memoria
-proprietà e utilizzo delle risorse

i processi formano un albero e ciascuno punta al padre.

Diapositiva 28
PROCESS TABLE
raggruppa i puntatori ai vari pcb.
MEMORY TABLES
fanno sì che a ciascun processo siano assegnate zone di memoria principale e secondaria (che il processo può usare per fare swapping). ogni regione di memoria ha degli attributi caratterizzanti in temini di accessi: una certa zona può essere acceduta in sola lettura o in lettura/scrittura.
I/O TABLES
memorizzano informazioni in relazione ai dispositivi che possono essere assegnati ad un processo: bisogna sapere se stiamo utilizzando un dispositivo, se siamo in attesa di input o di output, ecc...

Diapositiva 30
allocation of secondary memory to processes: eventualmente utilizzata per lo swap.

Diapositiva 34
Passaggio da modalità utente a modalità kernel. Tipicamente è causato da un interrupt o da una chiamata di sistema. Da interrupt: per esempio quando un dispositivo di IO ha terminato il suo compito, ed è necessario effettuare delle operazioni in kernel mode (lo switch è effettuato automaticamente dal processore - chiamata asincrona)
Quando un processo chiama una system call viene richiesto uno switch, se la cpu lo supporta lo effettua, altrimenti parte un'istruzione che SIMULA un interrupt. In Linux le system calls avvengono tipicamente in modo simulato
Da modalità kernel a modalità utente: Avviene in maniera più controllata perchè è il kernel che decide quando passare in user mode; in genere quando ha svolto tutti i compiti (es. schedulare processi...) allora passa in user mode e fa partire un certo processo.

Diapositiva 35
P1 è in esecuzione in user mode
mode switch -> kernel mode
process switch in kernel mode:
P1 stop P2 running

PROCESS DESCRIPTION AND CONTROL

mode switch -> user mode
P2 è in esecuzione in user mode

Diapositiva 36
i processi sono inseriti in apposite cose a seconda del loro stato: abbiamo in genere una coda di pronti in attesa di esecuzione, una coda di bloccati, di in esecuzione, ecc.

Diapositiva 38
Error or Exception: caso ad esempio di operazioni illecite, es divisione per zero.
memory fault: ad esempio se tento di scrivere in una cartella per la quale non è abilitato un permesso di scrittura

Diapositiva 39
Per alcune architetture delle parti sono trattate come processi (architetture microkernel).
Il kernel ha il suo spazio di memoria indipendente dagli altri processi: per eseguire il kernel bisogna riprogrammare le memory tables per accedere al kernel stesso. in questa architettura il mode switch costa quanto un process switch e la CPU deve passare alle aree di memoria del kernel, e quindi deve ricaricare le tabelle di memoria.
Molto poco efficiente.
Tipicamente usata negli OS attuali.
Quando si fa un mode switch le memory tables non vengono toccate, le strutture dati del kernel risiedono nello spazio di memoria dei processi.
Vantaggi. il mode switch costa molto poco; Svantaggi: la zona di memoria kernel non è accessibile ai processi, quindi lo spazio di indirizzamento disponibile è in realtà ridotto. i kernel UNIX funzionano così
il SO è trattato come un processo (o parte di esso) . Le system calls coinvolgono una comunicazione tra processi, comportando oneri maggiori rispetto ad una chiamata di sistema tradizionale; il numero di attività che vengono svolte fuori dai processi sono sostanzialmente di process switch e di comunicazione tra processi

Diapositiva 46
i processi del kernel vengono sicuramente sempre eseguiti in kernel mode mentre i processi utente no.
Dipende dall'approccio scelto dal preciso sistema operativo. Windows per esempio considera la maggior parte delle funzionalità come processi.

Diapositiva 47
le esecuzioni tutte interne al kernel diventano troppo legate tra loro, mentre se si considerano come processi separati sono più indipendenti l'uno dall'altro. Se un processo crasha basta ritirare su quel processo, ma il sistema intorno non subisce problemi.