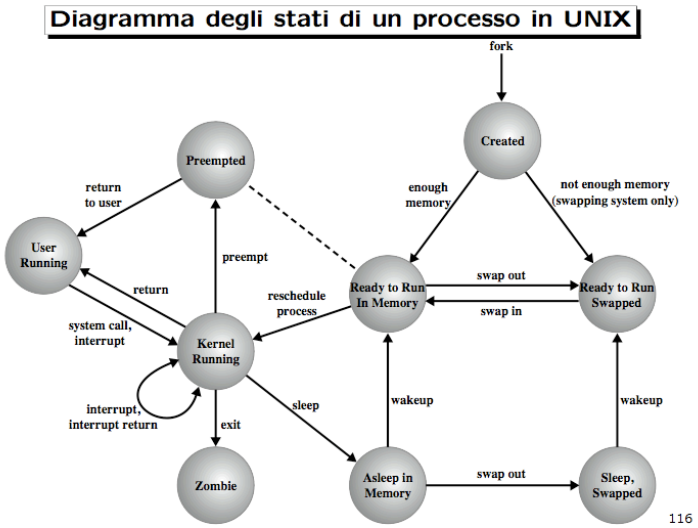
1. (a) Si descriva il diagramma degli stati di un processo in un sistema operativo generico.

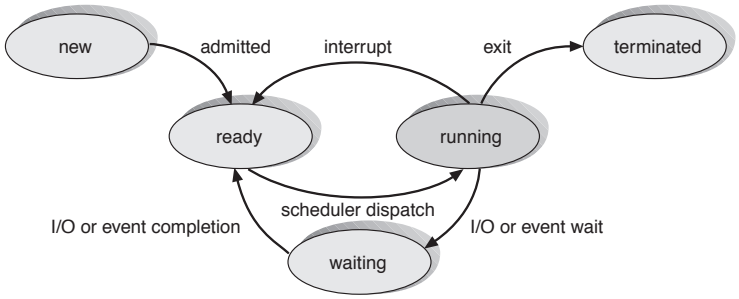
   (b) Si diano esempi di eventi che provocano il passaggio da uno stato ad un altro stato.

   (c) La seguente figura mostra il diagramma degli stati di un processo in Unix. Si spieghino le differenze con il diagramma generale discusso ai punti precedenti.



Risposta:

(a) Diagramma degli stati:



(b) Esempi di eventi che provocano il passaggio da uno stato ad un altro stato:

- Da new a ready: un nuovo processo viene allocato in coda ready.
- Da running a ready: in caso di scheduling della CPU con prelazione, un processo che passa da stato new in stato ready oppure da stato waiting a stato ready (per es. perché ha terminato un'operazione di I/O), può provocare il passaggio di un processo a minor priorità da stato running a stato ready.
- Da running a waiting: richiesta di esecuzione di un'operazione di I/O.

- Da waiting a ready: l'operazione di I/O è terminata.
- Da ready a running: un processo che passa da stato running a stato terminated o a stato waiting (in attesa di un evento, per es. il completamento di un'operazione di I/O), provoca il passaggio di un processo da ready a running. Questo tipo di passaggio di stato può avvenire anche per intervento dello scheduler di breve termine (per esempio in caso di schedulazione RR su interrupt del timer di sistema, qualora un processo abbia terminato il proprio quanto e rilasci la CPU ad un altro processo in coda ready).
- Da running a terminated: un processo termina la propria esecuzione.

(c) Nel caso del diagramma degli stati di un processo in Unix, le differenze con il diagramma generale discusso ai punti precedenti emergono dall'analisi degli stati:

- User running: esecuzione in modo utente,
- Kernel running: esecuzione in modo kernel,
- Ready to run, in memory: pronto per andare in esecuzione,
- Asleep in memory: in attesa di un evento; processo in memoria,
- Ready to run, swapped: eseguibile, ma swappato su disco,
- Sleeping, swapped: in attesa di un evento; processo swappato,
- Preempted: il kernel lo blocca per mandare un altro processo,
- Zombie: il processo non esiste più, si attende che il padre riceva l'informazione dello stato di ritorno.

In particolare il diagramma evidenzia l'esecuzione di codice in due modalità (utente e kernel) ed il fatto che i processi possono essere soggetti a swapping (per ridurre il grado di multiprogrammazione) ad opera dello scheduler di medio termine.

2. Sia data una variante dell'algoritmo di scheduling RR in cui gli elementi della coda dei processi pronti sono puntatori ai PCB.

   (a) Si descriva l'effetto dell'inserimento di due puntatori allo stesso processo nella coda dei processi pronti.

   (b) Si descrivano principali vantaggi e svantaggi di questo schema.

   (c) Si ipotizzi una modifica dell'algoritmo RR ordinario che consenta di ottenere lo stesso effetto senza duplicare i puntatori.

   Risposta:

   (a) Inserendo due puntatori allo stesso processo, quest'ultimo, nel caso in cui ad esempio i due puntatori siano consecutivi, si vedrebbe assegnare un tempo di CPU doppio rispetto agli altri processi. Infatti, scaduto il primo quanto di tempo, il primo puntatore al PCB verrebbe spostato in fondo alla coda, ma il secondo puntatore conferirebbe un ulteriore quanto di tempo al processo in questione. In generale quindi il processo beneficerebbe di un tempo di CPU doppio nel corso di ogni singola scansione della coda dei processi pronti.

(b) Un vantaggio derivante dall'utilizzo di questo sistema è quello di poter assegnare quanti di tempo diversi a seconda delle necessità di ogni singolo processo. Uno svantaggio potrebbe essere il rallentamento di alcuni processi a causa dell'inserimento "non accorto" dei puntatori "doppi" nella coda. Un ulteriore aspetto negativo è costituito dal fatto che nel caso di puntatori consecutivi allo stesso PCB ci sarebbe comunque uno spreco di tempo per l'esecuzione delle system call dovute agli interrupt del timer di sistema per gestire l'assegnazione dei quanti di tempo (tali chiamate di sistema potrebbero essere evitate in quanto il processo che va in esecuzione rimane semopre lo stesso).

(c) Per ottenere lo stesso effetto basterebbe assegnare un quanto di tempo "personalizzato" in base alle esigenze di ogni singolo processo.

3. Supponendo di avere un sistema con tre frame e sei pagine, addottando una politica di rimpiazzamento LRU, quanti page fault si verificheranno con la reference string seguente?

   0 1 5 2 3 2 5 1 2 4 1

   (Si assuma che i tre frame siano inizialmente vuoti.)

   Risposta: Si verificano 7 page fault come si vede dalla seguente simulazione:

0	1	5	2	3	2	5	1	2	4	1
	0	1	5	2	3	2	5	1	2	4
		0	1	5	5	3	2	5	1	2
			0	1	1	1	3	3	5	5
				0	0	0	0	0	3	3
									0	0
P	P	P	P	P			P		P	

4. Si illustrino brevemente le differenze fra I/O ad interrogazione ciclica (Programmed I/O), I/O guidato da interrupt (Interrupt-driven I/O) e trasferimento diretto in memoria (DMA).

   Risposta: Mentre con la modalità Programmed I/O (I/O a interrogazione ciclica) il processore manda un comando di I/O e poi attende che l'operazione sia terminata, testando lo stato del dispositivo con un loop busy-wait (polling), con la modalità Interrupt-driven I/O, una volta inviato il comando di I/O, il processo viene sospeso fintanto che non arriva un interrupt a segnalare il completamento dell'operazione. Durante la sospensione del processo, la CPU può mandare in esecuzione altri processi o thread. Di fondamentale importanza è il vettore di interrupt che consente di selezionare la routine di gestione opportuna per ogni tipo di interrupt. Ovviamente la prima modalità è efficiente soltando nel caso in cui la velocità del dispositivo di I/O sia paragonabile a quella della CPU. La modalità DMA richiede un controller DMA e funziona in questo modo: la CPU imposta i registri del controller DMA specificando il tipo di azione di I/O, l'indirizzo di memoria ed il conteggio di byte da trasferire. Poi i dati vengono trasferiti senza più richiedere l'intervento della CPU; infatti il controller del dispositivo di I/O riceve le richieste di lettura o scrittura

da parte del controller DMA a cui notifica il completamento dell'operazione una volta che ha trasferito il byte da/verso l'indirizzo di memoria corretto (specificato dal controller DMA). A questo punto il controller DMA incrementa l'indirizzo di memoria comunicandolo sul bus e decrementa il conteggio dei byte da traferire, ripetendo la richiesta di lettura o scrittura al controller del dispositivo fintanto che il conteggio dei byte non raggiungerà lo zero. Soltanto a questo punto verrà inviato un interrupt alla CPU che potrà far ripartire il processo sospeso. Siccome il controller DMA deve bloccare il bus per consentire i trasferimenti dal controller del dispositivo alla memoria, se anche la CPU ha bisogno di accedere al bus dovrà aspettare, venendo così rallentata.

5. Si spieghino le differenze fra architetture UMA e NUMA.

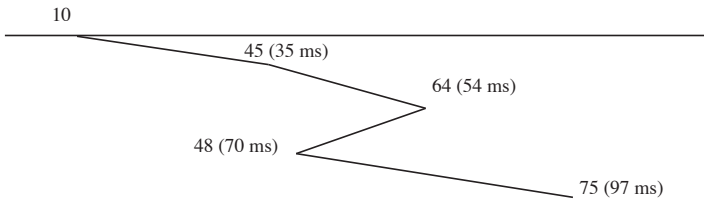
   Risposta: Un sistema multiprocessore UMA (Uniform Memory Access) è un sistema strettamente accoppiato che gode della proprietà che il tempo di accesso a qualunque locazione di memoria è sempre lo stesso (indipendentemente dal fatto che si faccia riferimento a memoria locale o di un altro nodo del sistema). Invece un sistema NUMA (Non Uniform Memory Access), pur rimanendo un sistema strettamente accoppiato, è caratterizzato dal fatto che l'accesso alla memoria locale è più veloce (da 2 a 15 volte) dell'accesso alla memoria remota. Ciò consente di ottenere sistemi scalabili con un gran numero di CPU.

6. Si consideri un disco gestito con politica C-LOOK. Inizialmente la testina è posizionata sul cilindro 10, ascendente; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 64, 45, 48, 75, rispettivamente agli istanti 0 ms, 30 ms 40 ms, 70 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite nell'ordine seguente: 45, 64, 48, 75.



   2. Il tempo di attesa medio per le quattro richieste in oggetto è dato da $\frac{(35-30)+(54-0)+(70-40)+(97-70)}{4} = \frac{5+54+30+27}{4} = \frac{116}{4} = 29ms.$

Il punteggio attribuito ai quesiti è il seguente: 3, 3, 3, 2, 2, 2, 4, 4, 3, 3, 2 (totale: 31).