Sistemi Operativi e Laboratorio, Prova del 9/9/2014

Nome: _____ Cognome: _____ Matricola:_____ fila:___ posto: ___

Esercizio 1 (5 punti)

Si consideri un processore che dispone dei registri speciali PC (program counter) e PS (program status), dello stack pointer SP e dei registri generali R1 e R2. In stato utente, ogni processo dispone di uno stack ad uso generale e di uno stack riservato per gestire le upcall chiamato *Signal Stack* (per semplicità assumiamo che ogni processo abbia un unico thread).

Per notificare una upcall ad un processo, il nucleo salva il program counter, lo stack pointer del processo e tutti i registri generali nel *Signal Stack*, quindi modifica lo stack pointer per puntare al Signal Stack, e il program counter per puntare alla procedura di gestione della upcall. La upcall si conclude ripristinando i registri generali dal Signal Stack, e quindi con l'istruzione RETU che ripristina PC e SP sempre dal Signal Stack.

Al tempo t, il nucleo riceve la richiesta di notifica di una upcall al processo P che corrisponde alla funzione di gestione che inizia all'indirizzo 3100 (nello spazio di memoria del processo). In questo istante di tempo, il signal stack punta alla locazione 2A19, il processo è in stato di pronto e il contenuto dei suoi registri speciali e generali è conservato nel suo descrittore come mostrato in tabella.

DESCRITTORE DI P	
Stato	Pronto
PC	5F80
PS	16F2
SP	FA00
R1	56A9
R2	52BE

Mostrare:

a) Il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di gestione della upcall;
b) Il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione RETU con la quale termina la upcall;
c) Il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la RETU.

Soluzione

a) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di gestione della upcall:

DESCRITTORE DI P	
Stato	
PC	
PS	
SP	
R1	
R2	

SIGNAL STACK DI P	
2A19	
2A18	
2A17	
2A16	
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	
R1	
R2	

PROCESSORE: Registri speciali e stato							
PC			PS			Stato	

b) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione RETU con la quale termina la upcall:

DESCRITTORE DI P	
Stato	
PC	
PS	
SP	
R1	
R2	

SIGNAL STACK DI P	
2A19	
2A18	
2A17	
2A16	
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	
R1	
R2	

PROCESSORE: Registri speciali e stato							
PC			PS			stato	

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la RETU.

DESCRITTORE DI P	
Stato	
PC	
PS	
SP	
R1	
R2	

SIGNAL STACK DI P	
2A19	
2A18	
2A17	
2A16	
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	
R1	
R2	

PROCESSORE: Registri speciali e stato							
PC			PS			stato	

Soluzione

a) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di gestione della upcall:

DESCRITTORE DI P	
Stato	Esecuzione
PC	5F80
PS	16F2
SP	FA00
R1	56A9
R2	52BE

SIGNAL STACK DI P	
2A19	5F80
2A18	FA00
2A17	56A9
2A16	52BE
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	2A15
R1	??
R2	??

PROCESSORE: Registri speciali e stato							
PC	3100		PS	16F2		Stato	Utente

b) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione RETU con la quale termina la upcall:

DESCRITTORE DI P	
Stato	Esecuzione
PC	5F80
PS	16F2
SP	FA00
R1	56A9
R2	52BE

SIGNAL STACK DI P	
2A19	5F80
2A18	FA00
2A17	
2A16	
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	2A17
R1	56A9
R2	52BE

PROCESSORE: Registri speciali e stato							
PC	3100+??		PS	16F2		stato	Utente

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la RETU.

DESCRITTORE DI P	
Stato	Esecuzione
PC	5F80
PS	16F2
SP	FA00
R1	56A9
R2	52BE

SIGNAL STACK DI P	
2A19	
2A18	
2A17	
2A16	
2A15	
2A14	

REGISTRI SP, R1, R2	
SP	FA00
R1	56A9
R2	52BE

PROCESSORE: Registri speciali e stato							
PC	5F80		PS	16F2		stato	Utente

Sistemi Operativi e Laboratorio, Prova del 9/9/2014

Esercizio 2 (5 punti)

In un File System UNIX semplificato gli i-node contengono 5 indirizzi di blocchi diretti e 2 indirizzi indiretti, rispettivamente per l'indirizzamento indiretto semplice e per l'indirizzamento indiretto doppio.
I blocchi hanno ampiezza di 2048 byte (= 2^{11} byte) e gli indici di blocco sono codificati con 2 byte.
Si consideri un file la cui lunghezza corrente è di 100.000 byte e il cui i-node contiene i seguenti puntatori:

ind	0	1	2	3	4	5	6
Blocco fisico	900	901	902	903	904	905	906

Alcuni frammenti dei blocchi 905, 906, 907 sono riportati nel seguito.
Blocco fisico 905:

Indice nel blocco	0	1	2	3	4	5	6	…
Blocco fisico	600	606	607	608	620	621	622	…

Blocco fisico 906:

Indice nel blocco	0	1	2	3	4	5	6	…
Blocco fisico	907	908	909	910	930	931	932	…

Blocco fisico 907:

Indice nel blocco	0	1	2	3	4	5	6	…
Blocco fisico	599	598	597	596	595	594	593	…

Si chiede:

1) la massima capacità (espressa in byte) dei dischi indirizzabili da questo File System;
2) Il numero di indirizzi contenuti in ciascun blocco indiretto, di primo o di secondo livello;
3) La massima dimensione teorica (espressa in byte) di un file in questo File System;
4) Quali blocchi bisogna leggere per leggere il carattere 16500 del file (ipotizzando che l'i-node sia già caricato in memoria)
5) Quali blocchi bisogna scrivere per modificare il carattere 2107400 del file (ipotizzando che l'i-node sia già caricato in memoria)

Soluzione

1) Possono essere indirizzati 2^{16} blocchi di 2^{11} byte ; la massima capacità del disco è di 2^{27} byte (128 Mbyte);
2) Ogni blocco indiretto contiene 2^{11} /2 = 1024 indirizzi ;
3) La massima dimensione (teorica) di un file è pari a 5+ 2^{10} + 2^{20} blocchi → 10KB + 2 MB+ 4 GB;
4) Il blocco che contiene il carattere 16500 è: 16500 **mod** 2048 = 8, che è indicizzato dal 4' puntatore (608) del blocco indice indiretto singolo 905. Pertanto **i blocchi da leggere sono il 905 e il 608**.
5) Il blocco che contiene il carattere 2107400 è: 2107400 **mod** 2048 = 1029, che è indicizzato dal 1' puntatore (599) del blocco indice indiretto doppio 907. Pertanto i blocchi da leggere sono il 906 e il 907 e **il blocco da scrivere è il 599**.

Sistemi Operativi e Laboratorio, Prova del 9/9/2014

Soluzione

1) La massima capacità disco è di _____ byte = _____ Mbyte;
2) Ogni blocco indiretto contiene _____ indirizzi ;
3) La massima dimensione (teorica) di un file è pari a _____ blocchi, pari a _____Byte;
4) Il blocco logico che contiene il carattere 16500 è il _____. I blocchi da leggere sono :_____.
5) Il blocco che contiene il carattere 2107400 è il _____. I blocchi da scrivere sono :_____.

Esercizio 3 (5 punti)

Un disco con 2 facce, 800 settori per traccia con indici nell'intervallo [0, 800) e 3000 tracce per faccia con indici nell'intervallo [0, 3000), ha un tempo di seek proporzionale al numero di tracce attraversate e pari a 0,02 ms per ogni traccia. Il periodo di rotazione è di 8 msec: conseguentemente il tempo impiegato per percorrere un settore è di 0,01 msec.
A un certo tempo (convenzionalmente indicato come t= 0) termina l'esecuzione dei comandi sulla traccia 860 e sono pervenute, nell'ordine, le seguenti richieste di lettura o scrittura:

- traccia 1200, faccia 0, settore 100
- traccia 630, faccia 1, settore 120
- traccia 1400, faccia 1, settore 5
- traccia 1400, faccia 1, settore 790
- traccia 420, faccia 0, settore 55

Inoltre, al tempo 24 arrivano tre ulteriori comandi di lettura:

- traccia 220, faccia 0, settore 60
- traccia 220, faccia 0, settore 5
- traccia 912, faccia 0, settore 60

Calcolare il tempo necessario per eseguire tutte queste operazioni supponendo che si adotti la politica di scheduling Shortest Seek Time First.
Il tempo di esecuzione di ogni operazione è uguale alla somma dell'eventuale tempo di *seek*, del ritardo rotazionale (tempo necessario per raggiungere il settore indirizzato) e del tempo di percorrenza del settore indirizzato. Per semplicità si assume che dopo un'operazione di *seek* le teste di lettura/scrittura impieghino un tempo pari al ritardo rotazionale per posizionarsi sul settore al quale è relativo il prossimo comando di lettura/scrittura. I comandi pendenti per lo stesso cilindro vengano eseguiti nell'ordine che minimizza il ritardo rotazionale.
Il controllore è dotato di sufficiente capacità di buffering ed è sempre in grado di accettare senza ritardo i dati letti dal disco o quelli da scrivere sul disco.

Soluzione

op. su cilindro:	630	settore:	120						
inizio:	0	seek:	4,6	rotazione:	8	percorrenza:	0,01	fine:	12,61
op. su cilindro:	420	settore:	55						
inizio:	12,61	seek:	4,2	rotazione:	8	percorrenza:	0,01	fine:	24,82
op. su cilindro:	220	settore:	5						
inizio:	24,82	seek:	4	rotazione:	8	percorrenza:	0,01	fine:	36,83
op. su cilindro:	220	settore:	60						
inizio:	36,83	seek:	0	rotazione:	0,54	percorrenza:	0,01	fine:	37,38
op. su cilindro:	912	settore:	60						
inizio:	37,38	seek:	13,84	rotazione:	8	percorrenza:	0,01	fine:	59,23
op. su cilindro:	1200	settore:	100						
inizio:	59,23	seek:	5,76	rotazione:	8	percorrenza:	0,01	fine:	73
op. su cilindro:	1400	settore:	790						
inizio:	73	seek:	4	rotazione:	8	percorrenza:	0,01	fine:	85,01
op. su cilindro:	1400	settore:	5						
inizio:	85,01	seek:	0	rotazione:	0,14	percorrenza:	0,01	fine:	85,16

Soluzione

op. su cilindro:		settore:			
inizio:		seek:	rotazione:	percorrenza:	fine:

op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:
op. su cilindro:	settore:			
inizio:	seek:	rotazione:	percorrenza:	fine:

Esercizio 4 (3 punti)

In un sistema multithreading, dove i thread sono realizzati a livello del nucleo e i processi sono unicamente contenitori di risorse, la gestione del processore avviene con una politica multi-level feedback queue (o a code multiple), che combina la politica a priorità con quella Round Robin, che si applica ai thread di uguale priorità. La politica prevede il prerilascio. I thread si sincronizzano con i meccanismi dello standard POSIX. Immediatamente prima del tempo *T* sono presenti i seguenti thread:

Processo 1: - T11, con priorità 4, in esecuzione;
- T12, con priorità 5, sospeso su *cond1*;
- T13, con priorità 1, pronto

Processo 2: - T21, con priorità 5, sospeso su *mutex2*;
- T22, con priorità 2, pronto;
- T23, con priorità 4, pronto

Processo 3: - T31, con priorità 2, sospeso su *mutex1*;
- T32, con priorità 3, pronto.

Si chiede:

1) la composizione delle code pronti per le classi di priorità 1, 2, 3, 4 e 5 immediatamente prima del tempo *T*;
2) Assumendo che a partire dal tempo *T* si verifichi la sequenza di operazioni riportata nella tabella dello schema di soluzione, si chiede per ciascuna operazione quale thread è in esecuzione e la composizione delle code pronti immediatamente dopo l'operazione.

Utilizzare la colonna NOTE per annotare la sospensione e la riattivazione dei thread.

Soluzione

1) Stato immediatamente prima di T:

Thread in esecuzione	CodaPronti5	CodaPronti4	CodaPronti3	CodaPronti2	CodaPronti1
T11	→∅	→T23 →∅	→T32→∅	→T22→∅	→T13→∅

2) stato risultante dalle operazioni:

sequenza di operazioni:	In esec.	Coda 5	Coda 4	Coda 3	Coda 2	Coda 1	note
eseguita pthread_cond_wait(&cond1)	T23	→∅	→∅	→T32→∅	→T22→∅	→T13→∅	T11 si sospende su cond1
eseguita pthread_mutex_unlock(&mutex2)	T21	→∅	→T23 →∅	→T32→∅	→T22→∅	→T13→∅	Riattivato T21
eseguita pthread_mutex_lock(&mutex1)	T23	→∅	→∅	→T32→∅	→T22→∅	→T13→∅	T21 sospeso su mutex1
Scade quanto di tempo	T23	→∅	→∅	→T32→∅	→T22→∅	→T13→∅	T23 resta in esecuzione
eseguita pthread_mutex_lock(&mutex2)	T32	→∅	→∅	→∅	→T22→∅	→T13→∅	T23 sospeso su mutex2
eseguita pthread_cond_wait(&cond1)	T22	→∅	→∅	→∅	→∅	→T13→∅	T32 sospeso su cond1
eseguita pthread_cond_signal(&cond1)	T12	→∅	→∅	→∅	→T22→∅	→T13→∅	Riattivato T12

Soluzione

1) Stato immediatamente prima di T:

Thread in esecuzione	CodaPronti5	CodaPronti4	CodaPronti3	CodaPronti2	CodaPronti1

2) stato risultante dalle operazioni:

sequenza di operazioni:	In esec.	Coda 5	Coda 4	Coda 3	Coda 2	Coda 1	note
eseguita pthread_cond_wait(&cond1)							
eseguita pthread_mutex_unlock(&mutex2)							
eseguita pthread_mutex_lock(&mutex1)							
Scade quanto di tempo							
eseguita pthread_mutex_lock(&mutex2)							
eseguita pthread_cond_wait(&cond1)							
eseguita pthread_cond_signal(&cond1)							

Esercizio 5 (3 punti)

In un sistema che gestisce la memoria con segmentazione a domanda, l'indirizzo logico è di 26 bit, dei quali i primi 6 bit codificano l'indice di segmento e i restanti 20 bit definiscono l'offset. Ogni elemento della tabella dei segmenti è formato da 8 byte, dei quali 1 byte contiene indicatori utili al gestore della memoria, 4 byte contengono il valore base, e i restanti 3 byte contengono il valore limite.

Si chiede:

1. Il massimo numero di segmenti di un processo;
2. La massima dimensione di un segmento (in byte);
3. La massima dimensione della tabella dei segmenti (in byte);
4. Il massimo valore possibile per l'indirizzo di origine di un segmento;

Soluzione

1. Massimo numero di segmenti di un processo: 2^6 segmenti = 64 segmenti
2. Massima dimensione di un segmento (in byte): 2^{20} byte = 1 Mbyte
3. Massima dimensione della tabella dei segmenti (in byte): 64*8= 512 byte
4. Massimo valore possibile per l'indirizzo di origine di un segmento: 2^{32}-1

Esercizio 6 (3 punti)

In un sistema che gestisce la memoria con paginazione, sono presenti i processi A, B e C. Lo stato di occupazione della memoria al tempo 20 è descritto dalla seguente tabella, dove per ogni blocco si specifica nell'ordine: il processo a cui appartiene la pagina caricata, l'indice della pagina e il tempo al quale è avvenuto l'ultimo riferimento alla pagina stessa. Ad esempio, nel blocco 11 è caricata la pagina 15 del processo A, il cui ultimo riferimento è avvenuto al tempo 8.

B,5 4	C,1 2	C,5 9		A,5 12		B,8 7	A,10 2	A,12 7	C,9 11	C,11 8	A,15 8	A,16 9	B,11 5	A,20 4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Per la gestione della memoria si utilizza un algoritmo di sostituzione LRU locale. Il working set assegnato al processo A ha dimensione 6; analogamente quelli assegnati ai processi B e C hanno dimensione 5.

Quali pagine vengono scaricate dalla memoria e caricate in memoria al tempo t= 21, 22 e 23 nelle seguenti ipotesi (alternative):

(a) Il processo A riferisce la pagina 2 al tempo 21, la pagina 3 al tempo 22 e la pagina 4 al tempo 23;
(b) Il processo A riferisce la pagina 10 al tempo 21, la pagina 20 al tempo 22 e la pagina 8 al tempo 23;
(c) Il processo B riferisce la pagina 5 al tempo 21, la pagina 6 al tempo 22 e la pagina 10 al tempo 23;
(d) Il processo B riferisce la pagina 1 al tempo 21, la pagina 2 al tempo 22 e la pagina 3 al tempo 23;
(e) Il processo C riferisce la pagina 0 al tempo 21, la pagina 9 al tempo 22 e la pagina 4 al tempo 23;
(f) Il processo C riferisce la pagina 1 al tempo 21, la pagina 2 al tempo 22 e la pagina 4 al tempo 23;

Soluzione

(a)

t=21: scaricata 10; caricata 2 nel blocco 7;

t=22: scaricata 20; caricata 3 nel blocco 14;

t=23: scaricata 12; caricata 4 nel blocco 8;

(b)

t=21: scaricata -; caricata - nel blocco -;

t=22: scaricata -; caricata - nel blocco -;

t=23: scaricata 12; caricata 8 nel blocco 8;

(c)

t=21: scaricata -; caricata - nel blocco -;

t=22: scaricata -; caricata 6 nel blocco 3;

t=23: scaricata -; caricata 10 nel blocco 5;

(d)

t=21: scaricata -; caricata 1 nel blocco 3;

t=22: scaricata -; caricata 2 nel blocco 5;

t=23: scaricata 5; caricata 3 nel blocco 0;

(e)

t=21: scaricata -; caricata 0 nel blocco 3;

t=22: scaricata -; caricata - nel blocco -;

t=23: scaricata 1; caricata 4 nel blocco 1;

(f)

t=21: scaricata -; caricata - nel blocco -;

t=22: scaricata -; caricata 2 nel blocco 3;

t=23: scaricata 11; caricata 4 nel blocco 10;

Soluzione

(a)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

(b)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

(c)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

(d)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

(e)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

(f)

t=21: scaricata _____; caricata _____ nel blocco _____;

t=22: scaricata _____; caricata _____ nel blocco _____;

t=23: scaricata _____; caricata _____ nel blocco _____;

Esercizio 7 (2 punti)

Dire quali delle seguenti affermazioni sono vere o false:

Se il sistema è in uno stato sicuro non si può verificare stallo, qualunque sia l'ordine di richiesta e assegnazione delle risorse	
Se il sistema è in uno stato non sicuro, è inevitabile lo stallo	
Alcuni processi in stallo possono essere in stato di *pronto*	
Se due processi sono in stallo per aver richiesto le risorse A e B, allora A e B sono non prerilasciabili	

Soluzione

Se il sistema è in uno stato sicuro non si può verificare stallo, qualunque sia l'ordine di richiesta e assegnazione delle risorse	FALSO
Se il sistema è in uno stato non sicuro, è inevitabile lo stallo	FALSO
Alcuni processi in stallo possono essere in stato di *pronto*	FALSO
Se due processi sono in stallo per aver richiesto le risorse A e B, allora A e B sono non prerilasciabili	VERO

Esercizio 8 (2 punti)

In un sistema simile a UNIX con interfaccia POSIX e scheduling a priorità, al tempo t sono presenti i (soli) processi P1, P2, P3 e P4, di priorità rispettivamente 2,4,3,3. I processi P1, P2 e P3 sono sospesi sulla variabile di condizione *condx*, mentre il processo P4 è in esecuzione.

Sempre al tempo t, il processo P4 esegue un'operazione di broadcast sulla variabile di condizione *condx*. Dire il contenuto delle code pronti, della coda associata a *condx*, e indicare il processo in esecuzione al termine dell'operazione.

Coda pronti priorità 1:	
Coda pronti priorità 2:	
Coda pronti priorità 3:	
Coda pronti priorità 4:	
Coda di *condx*:	
Processo in esecuzione:	

Soluzione

Coda pronti priorità 1:	-
Coda pronti priorità 2:	P1
Coda pronti priorità 3:	P3 -> P4
Coda pronti priorità 4:	-
Coda di *condx*:	-
Processo in esecuzione:	P2

Esercizio 9 (2 punti)

Dire quali delle seguenti affermazioni sono vere o false:

La chiamata di sistema comporta automaticamente il passaggio in stato supervisore	
La chiamata di sistema è equivalente ad un'invocazione di procedura o funzione	
Se i thread sono implementati a livello del nucleo, la commutazione di contesto tra due thread dello stesso processo deve essere svolta in stato supervisore	

Se i thread sono implementati a livello del nucleo, la commutazione di contesto tra due thread di processi diversi deve essere svolta in stato supervisore	

Soluzione

La chiamata di sistema comporta automaticamente il passaggio in stato supervisore	VERO
La chiamata di sistema è equivalente ad un'invocazione di procedura o funzione	FALSO
Se i thread sono implementati a livello del nucleo, la commutazione di contesto tra due thread dello stesso processo deve essere svolta in stato supervisore	VERO
Se i thread sono implementati a livello del nucleo, la commutazione di contesto tra due thread di processi diversi deve essere svolta in stato supervisore	VERO