SISTEMI OPERATIVI – 16 febbraio 2015
corso A nuovo ordinamento
e parte di teoria del vecchio ordinamento indirizzo SR

Cognome:_____ **Nome:**_____
Matricola:_____

1. Ricordate che non potete usare calcolatrici o materiale didattico, e che potete consegnare al massimo tre prove scritte per anno accademico.
2. Gli studenti a cui sono stati riconosciuti i 3 cfu di "linguaggio C" devono rispondere solo alle domande delle parti di teoria e di laboratorio Unix, e consegnare entro 1 ora e trenta minuti.
3. Gli studenti del vecchio ordinamento, indirizzo SR, devono rispondere solo alle domande della parte di teoria, e devono consegnare entro 1 ora.

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (5 punti)

a) Si consideri il problema dei produttori e consumatori, con buffer limitato ad *m* elementi, dove i codici del generico produttore e del generico consumatore sono riportati qui di seguito. Inserite le opportune operazioni di wait e signal necessarie per il corretto funzionamento del sistema, indicando anche i semafori necessari ed il loro valore di inizializzazione.

semafori necessari con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore full = 0;
semaphore empty = m;

"consumatore"	*"produttore"*
repeat	**repeat**
	<produci dato>
wait(full)	*wait(empty)*
wait(mutex)	*wait(mutex)*
<preleva dato dal buffer>	
	<inserisci dato nel buffer>
signal(mutex)	*signal(mutex)*
signal(empty)	*signal(full)*
<consuma dato>	
forever	**forever**

b) si supponga la presenza di m+1 produttori che cercano di inserire "contemporaneamente" nel buffer un elemento, mentre nessun consumatore ha ancora iniziato le sue operazioni. Cosa succede a ciascuno degli m+1 produttori?

L'*m*+1-esimo produttore si addormenta sul semaforo *empty*, mentre i precedenti *m* produttori riescono a superarlo. Di questi, *m*−1 si addormentano su *mutex*, mentre un produttore lo supera e inserisce un elemento nel buffer.

c) Vogliamo essere certi che la procedura A eseguita all'interno del processo P_A e la procedura B eseguita all'interno del processo P_B siano eseguite (non importa in quale ordine) prima che venga eseguita la procedura C all'interno del processo P_C. Riportate una soluzione a questo problema usando uno o più semafori, il loro valore di inizializzazione, e le usuali operazioni di wait e signal.

Semaphor *sync* = –1

P_A:	**P_B:**	**P_C:**
A	B	wait(*sync*)
signal(*sync*)	signal(*sync*)	C

d) Qual è la differenza fondamentale fra un gruppo di processi e un gruppo di peer thread?

I peer thread condividono lo spazio di indirizzamento.

ESERCIZIO 2 (5 punti)

Un sistema usa una paginazione a due livelli, e la tabella delle pagine esterna del processo più grande occupa esattamente un frame, e all'interno di una tabella delle pagine vengono usati 10 (esadecimale) bit per scrivere il numero di un frame. Si sa che in un indirizzo logico l'offset massimo all'interno di una pagina è 3FF.

a) Quanto sono grandi lo spazio di indirizzamento logico e fisico del sistema? (esplicitate i calcoli che fate).

Una pagina, e quindi un frame, sono grandi 2^{10}=1024 byte, e poiché vengono usati 2 byte (10 esadecimale bit) per scrivere il numero di un frame, la tabella delle pagine esterna più grande contiene 512 numeri di frame. La tabella delle pagine interna del processo più grande è dunque spezzata in 512 frame ognuno dei quali contiene 512 entry.
Lo spazio di indirizzamento logico è quindi di:$2^9 * 2^9 * 2^{10} = 2^{28}$ byte (256 megabyte).
Lo spazio di indirizzamento fisico è di $2^{16} * 2^{10} = 2^{26}$ byte (64 megabyte)

b) Supponendo che non si verifichino page fault, assumendo che il sistema sia dotato di un TLB perfetto con un hit rate medio del 90%, e che il tempo di accesso in RAM sia di 100 microsecondi, qual è il medium access time del sistema descritto sopra? (esplicitate i calcoli che fate)

mat = 0,9 * 100 + 0,1*(100+100+100) = 90 + 30 = 120 microsecondi

c. Nel sistema sopra riportato, è necessario stabilire una politica di allocazione (locale o globale che sia) dei frame? (motivate la vostra risposta)

Si, perché il sistema deve implementare la memoria virtuale.

d. In un sistema in funzione si osserva in un dato istante che la CPU ha una percentuale di utilizzo del 10%. Si decide allora di mandare in esecuzione un certo numero di nuovi processi CPU bound, ma la percentuale di utilizzo della CPU non aumenta. Come si può spiegare la cosa?

Il sistema è in thrashing.

ESERCIZIO 3 (4 punti)

Un hard disk ha la capienza di 2^{32} byte, ed è formattato in blocchi da 2048 byte.

a) Quanti accessi al disco sono necessari per leggere l'ultimo blocco di un file A della dimensione di 9.000 byte, assumendo che sia già in RAM il numero del primo blocco del file stesso e che venga adottata una allocazione concatenata dello spazio su disco? (motivate numericamente la vostra risposta)

5. Ogni blocco infatti memorizza 2045 byte di dati più 3 byte di puntatore al blocco successivo (infatti, $2^{32}/2^{11} = 2^{21}$), per cui sono necessari 5 blocchi per memorizzare l'intero file.

b) Quanto sarebbe grande, in megabyte, la FAT di questo sistema? (motivate numericamente la vostra risposta)

La FAT è un array con una entry per ciascun blocco dell'hard disk e che contiene il numero di un blocco, per cui: 2^{21} x 3 byte = 6 megabyte

c) Quali sono gli svantaggi nell'uso della FAT?

Per garantire un accesso efficiente ai file deve essere sempre tenuta in RAM, se viene persa si perdono tutte le informazioni sul file system, e deve quindi essere periodicamente salvata su disco.

d) si indichi quale configurazione RAID è opportuno scegliere nei seguenti casi:

il sistema RAID deve fornire un buon compromesso tra quantità di dati memorizzabili, velocità di accesso ai dati ed affidabilità (ossia tolleranza ai guasti): RAID 5

il sistema RAID deve memorizzare più dati possibili e fornire una buona velocità di accesso ad essi, l'affidabilità non è un requisito fondamentale: RAID 0

il sistema RAID deve avere la massima affidabilità e velocità di accesso ai dati: RAID 1

ESERCIZI RELATIVI ALLA PARTE DI UNIX (6 punti)

ESERCIZIO 1

(2 punti)

Il seguente segmento di codice intende creare tre processi figli. E' corretto? Motivare la risposta. Indicare inoltre cosa stampano i processi padre e figlio.

```
int k = 0;
while (k<3){
     int pid = fork();
     switch(pid){
          case -1: exit(1);
          case 0: printf("%d", k);
          default:print("%d", k);
     }
}
```

Il codice non è corretto, perché produce ben più di tre figli, per due motivi:

1. manca l'incremento della variabile k
2. ogni processo figlio generato dal padre genera a sua volta un altri processi figli (infiniti)

Tutti i processi, sia padre che figlio, stampano quindi il valore di k, che rimane fisso su 0.
La seguente versione di codice è invece corretta:

```
int k = 0;
while (k<3){
     int pid = fork();
     switch(pid){
          case -1: exit(1);
          case 0: printf("%d", k);
               exit(1);
          default:print("%d", k);
     }
     k++;
}
```

L'incremento di k consente la terminazione del loop.
La system call exit fa sì che il processo figlio, una volta stampato il valore di k, termini, e quindi non generi figli a sua volta.

ESERCIZIO 2

(2 punti)

Spiegare lo scopo della system call `msgrcv` indicando i suoi argomenti e il suo valore di ritorno.

Si vedano le slides del corso.

ESERCIZIO 3

(2 punti)

Nell'ipotesi che la directory corrente del terminale sia la home directory dell'utente Mario, e che non sia permesso modificare tale directory, scrivere i comandi da terminale necessari per ottenere i seguenti risultati (per ogni richiesta, i comandi devono essere dati su un'unica linea terminata da enter).

1) Elencare tutti i file con estensione .c che si trovano nella directory SOLAB e salvare il risultato in un file di nome sorgenti_c.txt nella directory TESTI. Entrambe del direcotry sono elencate nella home directory dell'utente Mario.

```
$ls SOLAB/*.c > TESTI/sorgenti_c.txt
```

2) La directoy TESTI contiene i seguenti file
E1.pdf, E2.pdf, E3.pdf, E12.pdf, E3.psf E1.cpp, E2f.pl, Ex.c
Scrivere il comando per **spostare** unicamente i file E1.pdf, E2.pdf, E3.pdf e E3.psf dalla directory TESTI alla directory SOLAB

```
$mv TESTI/E?.p[ds]f
```

ESERCIZI RELATIVI ALLA PARTE DI C (7 punti)

ESERCIZIO 1 (3 punti)

Si implementi la funzione con prototipo
`int all_letters(char * str, char * letters);`
che restituisce `TRUE` se tutte le lettere contenute in `letters` compaiono nell'ordine nella stringa `str`.
La funzione deve gestire opportunamente i casi di puntatori a `NULL` e stringhe vuote.
Definire opportunamente `TRUE` e `FALSE`.
Esempi:
se `str` fosse: agevolmente e `letters` fosse gvoent la funzione restituisce `TRUE`
se `str` fosse: agevolmente e `letters` fosse gvoetn la funzione restituisce `FALSE`
se `str` fosse: agevolmente e `letters` fosse gevoene la funzione restituisce `TRUE`

```
typedef enum {FALSE, TRUE} boolean;

int all_letters(char * str, char * letters){
     if (str==NULL || letters == NULL)
          return FALSE;
     int i=0, k=0;

     while (str[i]!='\0' && letters[k]!='\0'){
          if (str[i]==letters[k])
               k++;
          i++;
          printf("\n %c", letters[k]);
     }
     return letters[k]=='\0';
}
```

ESERCIZIO 2 (1 punto)

Data il seguente codice sorgente:

```
#include <stdio.h>
#include <string.h>

typedef struct
{
     char titolo[100];
     char autore[100];
     int annopubbl;
     float prezzo;
} libro;

void aumentaprezzolibro(libro *pl) {
     pl.prezzo = pl.prezzo * 1.1;
}

int main(void) {
     libro l = {"Guerra e Pace", "Federico Moccia", 2020, 16.00};
     strcpy(l->autore, "Lev Tolstoj");
     l.annopubbl = 2014;
     aumentaprezzolibro(&l);
     printf("prezzo: %g\n", l.prezzo);
}
```

indicare gli eventuali errori nell'accesso ai membri della struttura e correggerli.

```
#include <stdio.h>
#include <string.h>
typedef struct
{
     char titolo[100];
     char autore[100];
     int annopubbl;
     float prezzo;
} libro;

void aumentaprezzolibro(libro *pl) {
     pl->prezzo = pl->prezzo * 1.1;
}

int main(void) {
     libro l = {"Guerra e Pace", "Federico Moccia", 2020, 16.00};
     strcpy(l.autore, "Lev Tolstoj");
     l.annopubbl = 2014;
     aumentaprezzolibro(&l);
     printf("prezzo: %g\n", l.prezzo);
}
```

ESERCIZIO 3 (3 punti)

Data la struttura node definita come segue:

```
typedef struct node {
     int value;
     struct node * next;
} nodo;
typedef nodo* link;
```

implementare la funzione con prototipo

```
int extremes_are_multiple(link head);
```

che restituisce 1 se l'ultimo elemento della lista è un multiplo intero del primo e 0 altrimenti. Inoltre, la funzione deve restituire -1 se la lista è vuota.

```
int extremes_are_multiple (link head){
int ret_value = -1;

if(head != NULL) {
     int first_value;
     int last_value;

     first_value = head->value;
     while (head->next != NULL) {
          head = head->next;
     }
     last_value = head->value;
     if (last_value >= first_value && last_value%first_value==0)
          ret_value = 1;
     else
          ret_value = 0;
     }
     return ret_value;
}
```