Caselle riservate

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Sistemi Operativi

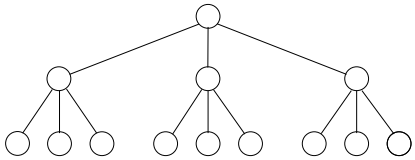
Compito d'esame

21 Febbraio 2015

Matricola _____ Cognome _____ Nome _____

Non si possono consultare testi, appunti o calcolatrici. Riportare i passaggi principali. L'ordine sarà oggetto di valutazione.
Durata della prova: 100 **minuti.**

1. Si scriva un programma C che ricevuti due valori interi sulla riga di comando, rispettivamente a e n, generi un albero di processi di altezza a e grado n. La figure riporta un esempio nel caso di $a = 2$ e $n = 3$.



Più nel dettaglio ogni nodo dell'albero è un processo. Il processo iniziale deve generare n processi figlio e terminare. La stessa cosa devono fare tutti i processi figli, generando così un numero di processi sulle foglie dell'albero pari a n^a. I processi sulle foglie devono tutti visualizzare il proprio PID e terminare.

2. Si descriva l'utilizzo dei segnali nel sistema operativo UNIX/Linux con relativi vantaggi e svantaggi. Si descrivano le system call `signal`, `kill`, `pause` e `alarm`. In particolare si indichi:
   - Se è possibile per un processo ignorare l'arrivo di un segnale.
   - Che cosa succede se un processo riceve un segnale durante l'esecuzione della funzione di gestione.
   - Se è possibile avere più funzioni di gestione associate a un segnale.
   - Se è possibile avere più segnali associati alla stessa funzione di gestione.
   - Come è possibile implementare la system call `alarm` tramite le system call `fork`, `signal`, `kill` e `pause`.

3. Si faccia riferimento alla sincronizzazione di processi mediante procedura `test-and-set`. Se ne illustrino le principali caratteristiche, riportandone il codice e il relativo protocollo di utilizzo. È possibile implementare una mutua esclusione senza starvation?

   Si supponga inoltre di avere a disposizione, al posto della `test-and-set` e della `swap`, la seguente funzione atomica:

```
int atomicIncrement (int *var) {
  int tmp = *var;
  *var = tmp + 1;
  return (tmp);
}
```

   Utilizzare tale funzione per scrivere le funzioni `init`, `lock` e `unlock` da inserire nel prologo e nell'epilogo di una sezione critica. *Suggerimento*: si utilizzino due variabili globali `ticketNumber` e `turnNumber`. La prima indica l'ordine di prenotazione per l'accesso alla sezione critica e la seconda il processo che ha il turno all'accesso. Gestire tali variabili con le funzioni di `init`, `lock` e `unlock`.

4. Il comando di shell "`ncal 02 2015`" visualizza il calendario del mese di febbraio dell'anno 2015 come evidenziato a sinistra dell'esempio successivo.

   All'interno del direttorio corrente tutti i file di estensione "`.cal`" contengono un insieme di date con il formato riportato al centro dell'esempio. Si scriva uno script BASH in grado di visualizzare (a video) l'elenco delle date presente su tutti i file di estensione "`.cal`" arricchito dal giorno delle settimana. Nel caso del file riportato al centro dell'esempio lo script deve visualizzare a video l'elenco riportato a destra.

```
    February 2015
Su  1  8 15 22
Mo  2  9 16 23
Tu  3 10 17 24
We  4 11 18 25
Th  5 12 19 26
Fr  6 13 20 27
Sa  7 14 21 28
```

```
8 9 2014
1 1 2015
2 2 2015
18 2 2015
...
```

```
8 9 2014 Mo
1 1 2015 Th
2 2 2015 Mo
18 2 2015 We
...
```

5. Si scriva uno script AWK in grado di sostituire le parole di un testo con un numero intero crescente inversamente proporzionale alla frequenza assoluta con cui ogni parola compare nel testo stesso. Il seguente esempio mostra l'azione dello script.

   A sinistra è riportato un possibile file di ingresso. Si supponga il file includa solo caratteri alfabetici. Caratteri alfabetici minuscoli e maiuscoli vanno considerati come equivalenti.

   Al centro è stato indicato il primo file di uscita. Esso riporta la frequenza assoluta (`freq`) di tutte le parole che compaiono nel file di ingresso con il relativo numero intero crescente (`codice`) ad esso associato. Tale codice è stato assegnato in ordine decrescente di frequenza, i.e., a frequenza maggiore corrisponde valore intero minore.

   A destra è riportato il secondo file di uscita. In esso tutte le parole del file di ingresso sono sostituite dal loro numero intero (`codice`) corrispondente.

```
esempio esempio
Esempio per script AWK
Script AWK
ESEMPIO awk
```

```
esempio freq=4 codice=1
per freq=1 codice=4
script freq=2 codice=3
awk freq=3 codice=2
```

```
1 1
1 4 3 2
3 2
1 2
```

   Il nome dei tre file può essere passato allo script sulla riga di comando o, in alternativa, assegnato a variabili dello script sulla riga di comando stessa. Si ricorda che il comando `str=toupper(str)` (`str=tolower(str)`) trasforma la stringa `str` in caratteri maiuscoli (minuscoli).

6. Si illustri l'*algoritmo del banchiere* riportandone la descrizione e lo pseudo-codice ottimizzato al caso di esattamente tre processi e una sola risorsa.

   Analizzando l'esempio successivo, con processi $(P_1, \ldots, P_5)$ e risorse $(R_1, \ldots, R_4)$, si indichi se lo stato è sicuro oppure non sicuro. Nel primo caso, si illustri l'ordine in cui i processi possono essere completati. Se arrivasse una richiesta di P_1 per $(1, 1, 2, 0)$, tale richiesta potrebbe essere soddisfatta?

Processo	Fine	Assegnate				Massimo				Necessità				Disponibilità			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	F	2	0	0	1	4	2	1	2					3	3	2	1
P_2	F	3	1	2	1	5	2	5	2								
P_3	F	2	1	0	3	2	3	4	6								
P_4	F	1	0	1	2	1	1	2	4								
P_5	F	1	1	3	2	3	6	6	5								