

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

Sistemi Operativi

Compito d'esame

27 Febbraio 2017

Matricola _____ Cognome _____ Nome _____

Docente: Quer Sterpone

Non si possono consultare testi, appunti o calcolatrici a parte i formulari distribuiti dal docente. Riportare i passaggi principali. L'ordine sarà oggetto di valutazione.

Durata della prova: 100 minuti.

1. Si descriva l'utilizzo dei segnali nel sistema operativo Unix/Linux con relativi vantaggi e svantaggi. Si descrivano in particolare le system call `signal` e `kill`.

Si riportino inoltre due esempi di gestione dei segnali. Il primo illustri come la system call `alarm` possa essere implementata tramite le system call `fork`, `signal`, `kill` e `pause`. Il secondo descriva come la system call `alarm` possa essere utilizzata da un processo che desidera continuare a svolgere i suoi compiti ma visualizzare dopo 10 secondi un preciso messaggio su standard output.

2. Si descrivano le caratteristiche principali delle *pipe* per la comunicazione e la sincronizzazione tra processi e se ne illustri l'utilizzo realizzando il seguente programma in linguaggio C.

Un processo P , dopo aver generato un figlio F , effettua le seguenti operazioni:

- (a) Genera un numero casuale intero r incluso nell'intervallo $[1, 3]$.
- (b) Se $r = 1$ legge da standard input un numero intero n e una stringa s , trasmette al figlio F su una pipe il carattere F , l'intero n , e la stringa s , e ritorna al punto (a).
- (c) Se $r = 2$ trasmette al figlio F il carattere X e ritorna dal punto (a).
- (d) Se $r = 3$ termina.

Il processo figlio F effettua le seguenti operazioni:

- (a) Legge i dati trasmessi sulla pipe (un carattere F seguito da altri dati, oppure un carattere X singolo).
- (b) Se il primo carattere ricevuto è una F , genera a sua volta un figlio e ritorna al punto (a). Il figlio (del processo F) si occupa di visualizzare n volte a distanza di 1 secondo la stringa s e termina ritornando come valore di terminazione il valore di n .
- (c) Se il primo carattere ricevuto è una X , raccoglie e visualizza su standard output i codici di terminazione di tutti i suoi figli (generati sino a quel momento) e termina.

La stringa s abbia lunghezza costante uguale a 20 caratteri e non contenga spazi. Si ricorda che il codice di terminazione di un figlio può essere catturato con la macro `WEXITSTATUS`. Si utilizzi l'espressione `((int)(rand()%3)+1)` per generare un numero casuale nell'intervallo $[1, 3]$.

3. Nello studio delle situazioni di stallo (deadlock), si chiarisca che cosa si intende per “condizioni necessarie”, illustrando singolarmente tali condizioni. Si dimostri inoltre (in maniera formale) come sia possibile evitare l’*attesa circolare*.

Siano dati infine i tre processi successivi utilizzando i semafori A, \dots, F tutti inizializzati a 1. Esiste una sequenza temporale di esecuzione di P_1, P_2 e P_3 tale per cui si crei una situazione di stallo? Motivare la risposta.

```
P1: while (1) {           P2: while (1) {           P3: while (1) {
    wait (D);              wait (C);              wait (A);
    wait (E);              wait (F);              wait (B);
    wait (B);              wait (D);              wait (C);
    printf ("P1\n");       printf ("P2\n");       printf ("P3\n");
    signal (D);            signal (C);            signal (A);
    signal (E);            signal (F);            signal (B);
    signal (B);            signal (D);            signal (C);
}                          }                          }
```

4. Per ottimizzare lo spazio su disco, si scriva uno script BASH in grado di:

- Ricevere 3 parametri sulla riga di comando, i.e., il nome di un utente, un numero intero e il nome di un file di log, verificandone la corretta ricezione.
- Elencare ricorsivamente tutti i file presenti nella home directory dell'utente di dimensioni maggiori al numero specificato (valore in MBytes) e per ognuno di essi chiedere all'utente se desidera:
 - D: (delete) cancellare il file.
 - Z: (gzip) comprimere il file (cancellando poi il file originale).

Lo script deve effettuare le operazioni richieste e, per ognuna di esse, appendere nel file di log una riga del tipo:
`type user date originalSize currentSize path`

in cui il campo:

- `type` è D nel caso di un file cancellato oppure Z nel caso di un file compresso.
- `user` indica il nome dell'utente proprietario del file.
- `date` indica la data di cancellazione e/o compressione del file (il comando `date` fornisce su standard output la data corrente in formato `aaaa-mm-gg`, e.g. `2017-02-27`).
- `originalSize` (e `currentSize`) indicano la dimensione (in bytes) dei file prima (e dopo) la cancellazione (compressione).
- `path` è il percorso assoluto del file rimosso (o compresso).

Per comprimere il file si utilizzi il comando `gzip <nomeFile>` che produce una versione del file con stesso nome e ulteriore estensione `gz`. Si ricorda che il comando `ls -l` produce in output righe del tipo:

```
permissions num_links owner_group owner_user bytesize last_access name
```

5. Scrivere uno script AWK che processi un file con lo stesso formato descritto nell'esercizio precedente, ovvero costituito da righe del tipo:

```
type user date originalSize currentSize path
```

Lo script deve calcolare la quantità di spazio su disco risparmiata in totale e per i singoli utenti. In particolare, per ogni utente, lo script deve fornire in output la quantità di spazio liberato (in Mbytes) mediante cancellazione, la quantità di spazio liberato (in Mbytes) mediante compressione e la quantità di spazio liberata (in Mbytes) totale. Inoltre lo script deve stampare le quantità di spazio liberate a livello di intero sistema, considerando la cancellazione e la compressione dei file di tutti gli utenti. Ad esempio, dato il seguente file

```
Z gianni 2017-01-16 739239936 89229201 /home/gianni/Video/vid.avi
D gianni 2017-01-16 1468033024 0 /home/gianni/dvd.iso
D paolo 2017-01-18 2990078103 0 /home/paolo/folder/archive.zip
Z paolo 2017-01-18 693704050 46246936 /home/paolo/Video/vid.mp4
```

lo script deve fornire un output del tipo:

```
gianni:  deleted=1400  compressed= 619  total=2019
paolo:   deleted=2851  compressed= 617  total=3468
overall: deleted=4251  compressed=1236 total=5487
```

6. Si consideri il seguente insieme di processi:

Processo	Tempo arrivo	Burst Time	Priorità
P ₀	0	22	5
P ₁	0	16	2
P ₂	15	19	4
P ₃	17	7	1
P ₄	25	15	3

Rappresentare mediante diagramma di Gantt l'esecuzione di tali processi utilizzando gli algoritmi di scheduling PS (Priority Scheduling), RR (Round Robin) e SRTF (Shortest Remaining Time First). Calcolare inoltre il tempo di attesa medio per ciascun processo.

Si osservi che la priorità maggiore è associata al valore di priorità inferiore e che per la strategia Round Robin il quantum temporale è di 10 unità di tempo.