Laboratorio di Sistemi Operativi
13 Settembre 2019
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (4 punti) Si scriva una pipeline che stampi a video il numero di utenti che abbiano la home directory all'interno del percorso `/home`.

   Suggerimento: si ricorra al file `/etc/passwd` dove ogni linea è una successione di campi separati dai due punti (`:`) e la home directory è rappresentata dal sesto campo. Ad esempio:

   `pippo:x:1017:1019:Pippo,,,:/home/fernandez:/bin/bash`

```
cat /etc/passwd | cut -d':' -f6 | grep '^/home' | wc -l
```

2. (6 punti) Si scriva uno script `home.sh` della shell che stampi a video il numero di sottodirectory visibili di primo livello (quindi non si deve scendere ricorsivamente nell'albero del filesystem) della directory `/home`.

```
1  count=0;
2
3  for i in /home/*
4  do
5     if test -d $i
6     then
7       count=$[$count+1]
8     fi
9  done
10
11 echo $count
12
13 exit 0
```

3. (10 punti) Si specifichi una struttura dati ricorsiva per implementare una lista concatenata di interi e si scriva un programma C che effettui le seguenti operazioni:

   1. legga dei numeri interi dallo standard input fino a quando non incontri il segnale di fine file (`EOF`);
   2. per ogni intero letto dal file allochi un nodo della lista e vi memorizzi il numero intero;
   3. alla fine stampi la lista di interi letti e memorizzati e la rimuova dalla memoria.

   Suggerimento: usare `scanf()` che restituisce come valore il numero di conversioni effettuate con successo o `-1` se incontra `EOF`.

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

struct node_t {
  int data;
  struct node_t *next;
};
```

```c
typedef struct node_t Node;

Node *insert(Node *p,int d);
void list(Node *p);
void delete(Node *p);

int main() {
  Node *head=NULL;
  int n;

  while(scanf("%d",&n)==1) {
    head=insert(head,n);
  }

  list(head);
  delete(head);
  return 0;
}

Node *insert(Node *p,int d) {
  Node *q=p;
  if(p==NULL) {
    p=(Node *)malloc(sizeof(Node));
    p->data=d;
    p->next=NULL;
  } else {
    while(q->next!=NULL) q=q->next;
    q->next=(Node *)malloc(sizeof(Node));
    q->next->data=d;
    q->next->next=NULL;
  }
  return p;
}

void list(Node *p) {
  if(p==NULL) {
    printf("_\n");
  } else {
    printf("%d -> ",p->data);
    list(p->next);
  }
}

void delete(Node *p) {
  Node *q;

  if(p!=NULL) {
    q=p->next;
    free(p);
    delete(q);
  }
}
```

4. (12 punti) Si scriva un programma C (`stampa_file.c`) che accetti su linea di comando il percorso di un file, generi un processo figlio e si metta in attesa della sua terminazione prima di terminare a

sua volta. Il processo figlio deve stampare a video il contenuto del file passato su linea di comando. La gestione degli errori (e.g., mancanza del parametro su linea di comando, numero di parametri errati ecc.) deve essere eseguita dal padre prima della generazione del processo figlio (che andrà in porto solo in caso di assenza di errori).

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(int argc, char **argv) {
  pid_t pid;
  FILE *f;
  char c;

  if(argc!=2) {
    fprintf(stderr,"wrong number of arguments!\n");
    return 1;
  }

  f=fopen(argv[1],"r");

  if(f==NULL) {
    fprintf(stderr,"fopen failed!\n");
    return 1;
  }

  pid=fork();

  switch(pid) {
    case -1:
      fprintf(stderr,"fork failed!\n");
      return 1;
    case 0:
        while((c=fgetc(f))!=EOF) {
          printf("%c",c);
        }
        fclose(f);

        return 0;
    default:
      waitpid(pid,NULL,0);
  }

  return 0;
}
```