UNIX - Programmation C

FILIÈRE ÉLECTRONIQUE, 1ère ANNÉE

2022-2023

Rémi Giraud - remi.giraud@enseirb-matmeca.fr
Frédéric Chatrie, Florent Grélard, Jean-Charles Henrion, Édern Le Bot

Examen

Nom - Prénom : _____

1 Questions de cours (5 points)

- Quelle est la valeur de la variable `x` pour les calculs suivants ?

  ```
  float x = (float) (1/2);   _____
  float x = ((float)1)/2;    _____
  float x = 1/2;             _____
  float x = 1.0/2.0;         _____
  ```

- Quelle est la taille en octets d'une variable de type `int` ?

- Quelle est la fonction qui permet de calculer la longueur d'une chaîne de caractères ?
  Pour un tableau d'entiers quelconque ?

- Qu'est-ce qu'une erreur de segmentation ? Que se passe-t-il quand elle se produit ? Le programme compile-t-il ? S'exécute-t-il ?

- Est-il possible de déclarer dans une fonction, une variable qui porte le même nom qu'une variable globale ?

- Quelle est la taille en octets d'un pointeur vers un `short int` ? (sur une architecture 64 bits classique)

2 Analyse de codes (5 points)

2.1 Fonction mystère (3 points)

Pour chaque fonction, décrire en maximum une phrase ce que fait la fonction. À la limite, une phrase supplémentaire par paramètre pour en expliquer l'utilité. Ne pas décrire linéairement le programme et les instructions mais résumer le but de chaque fonction (par ex. cette fonction renvoie la valeur absolue d'un entier x donné en paramètre).

```
int blabla(char bli, const char* blou) {
  int uob = 0;
  while(blou[uob])
    uob++;
  for (int plop=0; plop<uob; plop++)
    if (blou[plop] == bli)
      return plop;
  return -1;
}
```

```
float francis(float* star, int michel) {
  float johnny = 0.0;
  int alain = michel ;
  while (alain--)
    johnny += star[alain]*star[alain];
  return sqrt(johnny);
}
```

```
void gus(int* catoche, int didi, int* olive) {
*olive = *catoche;
for (int zed=1; zed<didi; zed++)
if (*olive < *(catoche+zed))
*olive = *(catoche+zed);
}
```

2.2 Code bugué (2 points)

La fonction suivante compte le nombre de valeurs négatives dans un tableau de doubles. 4 erreurs se sont glissées dedans. Ré-écrire le programme en corrigeant les erreurs.

```
int count_negs(double *tab, int len) {
  float negs = 0;
  for (ind = 0; ind<=len; ind++)
    if (tab[ind] < 0)
      ++negs;
      printf("%lf\n", tab[ind]);
  return negs;
}
```

3 Programmation (10 points)

3.1 Valeur absolue (1 point)

Écrire une fonction qui retourne la valeur absolue d'un entier `int`.

3.2 Chaîne de caractères (2 points)

Écrire une fonction qui copie une chaîne de caractères source `src` dans une chaîne de caractères destination `dst` donnée également en paramètre. Cette fonction retourne un pointeur sur le 0 terminal de la destination. On supposera la chaîne `dst` capable d'accueillir `src`.

3.3 Conversion entier/binaire (2 points)

Écrire une fonction `itob` qui reçoit un `int` comme argument et qui renvoie une chaîne de caractères correspondant à la représentation binaire de ce nombre.
Exemple : Pour un `int` sur 4 octets de valeur 13 on aura : "00000000000000000000000000001101".
(Rappels : opérateurs de décalage de bits : $<<, >>$, opérateurs de comparaison bits-à-bits : $\&, |$)

3.4 Structures (5 points)

Dans cet exercice, nous allons créer un type “carnet d’adresses” et programmer un certain nombre de fonctions basiques permettant de le manipuler.

- Écrire une structure `contact` comprenant les champs `numéro d’identification` (un entier de type `int`), `nom, prénom, adresse, genre` et `téléphone` (chaînes de caractères).

  Existe-t-il une information qui peut être facilement représentée par une énumération `enum` ?

- Écrire une fonction `display_contact` permettant d’afficher les informations d’un contact donné en paramètre.

- Écrire une fonction `create_contact` permettant de saisir les informations sur un contact. La fonction ne prend pas d’argument, renvoie une structure `struct contact` et interagit avec un utilisateur par la fonction `scanf`.

  (On considérera que `scanf` est capable de récupérer une chaîne de caractères contenant des espaces, pour initialiser le champ adresse par exemple, même si ça n’est pas vrai en pratique).

- Créer une structure de type liste chaînée `lst_contact` vers des contacts. Cette structure contiendra logiquement un `contact` et un pointeur vers un nouvel élément de type `lst_contact`.

- Écrire une fonction `display_all_contacts` qui prend en argument une liste chaînée de type `lst_contact`, et parcourt cette liste pour afficher tous les contacts.

  On pourra faire appel à la fonction `display_contact`.

  On considérera la liste chaînée bien construite, c'est-à-dire que le champ pointeur d'un élément `lst_contact`, s'il n'est pas à `NULL`, emmène vers un nouvel élément valide. S'il vaut `NULL` c'est qu'on est sur le dernier élément de la chaîne.