Licence Informatique
Université Bordeaux I

Initiation à la Programmation
Devoir Surveillé Novembre 2002

L'épreuve dure 2 heures et aucun document n'est autorisé.
Les programmes sont exclusivement à rédiger en langage C - ansi. On respectera le mode d'indentation utilisé en cours. La notation tiendra compte de la correction des programmes mais surtout de la qualité du codage : lisibilité, maintenabilité, extensibilité et concision.
Barème indicatif : 3 - 3 - 9 - 5.

Exercice 1. Illustrer et préciser les différents emplois du mot réservé `static`.

Exercice 2. Écrire le texte que produirait l'exécution du programme suivant :

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

char *
f (char *a, int i)
{
  static int c = 0;
  c += i;
  return a + c;
}

int
main (void)
{
  char *a = "segmentation fault, core dumped";

  printf ("%c\n", *f (a, 5));
  printf ("%c\n", *f (a, 0));
  printf ("%c\n", *f (a, 8));
  *f (a, 0) = 'a';
  printf ("%c\n", *f (a, 0));

  return EXIT_SUCCESS;
}
```

Exercice 3. Écrire le programme correspondant à la commande

`parametre_le_plus -<option> <liste de paramètres>`

où option peut prendre les valeurs suivantes :
- `petit` : affiche le paramètre le plus petit au sens de `strcmp`.
- `grand` : affiche le paramètre le plus grand au sens de `strcmp`.

Exercice 4. Écrire le code de la fonction `unsigned long strtoul(char *nptr, char **endptr, unsigned int base)` dont le fonctionnement est précisé au verso.

La variable globale `errno` et la constante `ERANGE` sont déclarées dans `<errno.h>`; la constante `ULONG_MAX` est déclarée dans `<limits.h>`.

Simplifications : on utilisera - sans la définir - la fonction `int valeur(char c, int base)` qui retourne la valeur du caractère donné dans la base indiquée ou -1 si le caractère n'est pas un chiffre de la base. On ne traitera ni le cas de la base spéciale 0, ni les espaces en début de chaîne, ni les préfixes `0x` et `0`.

Correction

Exercice 1. Le mot réservé `static` à deux emplois :

1. Pour les objets de profondeur zéro (fonctions et variables globales), il signifie que la portée de la déclaration est limitée au module (fichier) où il est déclaré.
2. Pour les objets de profondeur supérieur (variables locales), static donne l'ordre au compliateur de placer cette variable locale dans le tas et non dans la pile. Tout comme les variables globales leur initialisation est définie de façon statique (à la compilation, on a donc affaire à une constante) et réalisée avant l'exécution du programme principale (jamais lors du premier appel) ; c'est pourquoi l'initialisation d'une variable statique ne peut se faire par appel de fonction.

Exercice 2.

```
n
n
f
erreur de segment
```

Exercice 3.

Cet exercice ressemble vraiment au programme Mesurer, la différence est que nous connaissons maintenant l'arithmétiques des pointeurs pouvons donc faire en sorte que les fonctions qui calcule le min et le max d'un tableau de chaînes soient réellement réutilisables.

.... Fichier principale ....

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

#include "chaine.h"

void
usage (char *argv0)
{
  fprintf (stderr, "usage : %s <-petit|-grand> <liste de parametres>", argv0);
  exit (EXIT_FAILURE);
}

int
main (int argc, char *argv[])
{
  char *s;

  if (argc < 3)
    usage (argv[0]);

  if (!strcmp (argv[1], "-petit"))
    s = chaine_min (argv + 2, argc - 2);
  else if (!strcmp (argv[1], "-grand"))
    s = chaine_max (argv + 2, argc - 2);
  else
    usage (argv[0]);

  printf ("%s\n", s);

  return EXIT_SUCCESS;
}
```

....Fichier chaine.h ....

```c
#ifndef CHAINE_H
#define CHAINE_H

#include <stdlib.h>

char *chaine_min (char *chaine[], size_t nb);

char *chaine_max (char *chaine[], size_t nb);

#endif
```

....Fichier chaine.c ....

```c
#include "chaine.h"

#include <stdlib.h>
#include <string.h>

char *
chaine_min (char *chaine[], size_t nb)
{
  size_t i;
  char *min;

  if (nb == 0)
    return NULL;

  min = chaine[0];
  for (i = 1; i < nb; i++)
    if (strcmp (chaine[i], min) < 0)
      min = chaine[i];

  return min;
}

char *
chaine_max (char *chaine[], size_t nb)
{
  size_t i;
  char *max;

  if (nb == 0)
    return NULL;

  max = chaine[0];
  for (i = 1; i < nb; i++)
    if (strcmp (chaine[i], max) > 0)
      max = chaine[i];

  return max;
}
```

Exercice 4.

Seules les lignes 35-36 et 60-74 étaient espérées.

```
   #include <assert.h>
2  #include <ctype.h>
   #include <errno.h>
4  #include <limits.h>
   #include <stdio.h>
6  #include <stdlib.h>

8  enum
   { MAX_BASE = 37 };
10


12 static int
   valeur (char c, int base)
14 {
     int v;
16
     if (c >= '0' && c <= '9')
18     v = c - '0';
     else if (c >= 'a' && c <= 'z')
20     v = c - 'a' + 10;
     else if (c >= 'A' && c <= 'Z')
22     v = c - 'A' + 10;
     else
24     return -1;

26   return v < base ? v : -1;
   }
28
   unsigned long int
30 strtoul (const char *nptr, char **endptr, int base)
   {
32   unsigned long resultat;
     int chiffre;
34
     assert (base >= 0 && base != 1 && base < MAX_BASE);
36   errno = 0;

38   while (isspace (*nptr))
       nptr++;
40
     if (*nptr == '+' || *nptr == '-')
42     nptr++;

44   if (base == 0)
       {
46       if (*nptr == '0')
           {
48           base = 8;
             nptr++;
50         }
         else if (*nptr == '0' && nptr[1] == 'x')
52         {
             base = 16;
54           nptr = nptr + 2;
           }
56       else
```

```c
        base = 10;
    }

  for (resultat = 0; (chiffre = valeur (*nptr, base)) != -1; nptr++)
    if (resultat > ULONG_MAX / base ||
        (resultat * base > ULONG_MAX - chiffre))
      {
        resultat = ULONG_MAX;   // on continue la lecture malgre le
        errno = ERANGE;         // depassement de capacite afin de
      }                         // correctement positionner nptr
    else
      resultat = resultat * base + chiffre;

  if (endptr != NULL)
    *endptr = nptr;

  return resultat;
}


/*********************** TEST *************************
int
main()
{
  char n[20];

  sprintf(n,"%uld", ULONG_MAX);

  printf("%u\n",strtoul("1234",NULL,10));


  printf("%u\n", ULONG_MAX);

  sprintf(n,"%u", ULONG_MAX-1);
  printf("%u\n", strtoul(n,NULL,10));

  sprintf(n,"%u", ULONG_MAX);
  printf("%u\n", strtoul(n,NULL,10));

  perror("tout va bien");

  printf("%u\n", strtoul("4294967296",NULL,10));

  perror("tout va presque bien");

  return EXIT_SUCCESS;
}
*****************************************************/

/********************** TEST **************************/
/* 1234 */
/* 4294967295 */
/* 4294967294 */
/* 4294967295 */
/* tout va bien: Success */
/* 4294967295 */
/* tout va presque bien: Numerical result out of range */
```