1. Que fait le programme suivant ?
2. A l'aide d'un schéma, représentez l'espace d'adressage du processus sous linux et ses différents segments.
3. Décrivez le code et son fonctionement en détail. Au fur et a mesure de vos explications indiquez pour chaque variable dans quelle partie de l'espace d'adressage elle ce situe. Indiquez également lorsque le tas est modifié.
4. La chaine de caractère str est mal construite. Que lui manque-t-il ?
5. Que ce passe-t-il si j'appel le programme de la manière suivante:

   convert 546213354863513268423132654

   Comment éviter ce problème ?

convert.c

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

#define E_NOT_VALID_CHAR -1
#define NB_BITS_OCTETS 8

typedef short int number;

char errorMsg[] = "Erreure non documentee\n";
int unUsedVariable; //variable innutile sauf pour la question 3

int convert(number* res, const char* chaine) {
    int i,length = 0;
    *res = 0;

    if(*chaine == '\0')
        return E_NOT_VALID_CHAR;

    while(chaine[length] != '\0')
        length++;
    length = length - 1;


    for(i=length; i>=0; i--) {
        if( (chaine[i] < '0') || (chaine[i] > '9') )
            return E_NOT_VALID_CHAR;

        *res += (chaine[i] - '0') * pow(10, length-i);
    }

    return 0;
}

char* getStr(number value) {
    number testBit;
```

```
37      int i, nbBits;
38
39      nbBits = sizeof(value) * NB_BITS_OCTETS;
40
41      char* str = (char*) malloc(nbBits);
42      if(str == NULL)
43          return NULL;
44
45      for(i=0; i<nbBits; i++) {
46          testBit = pow(2,i);
47          if( (testBit & value) == testBit )
48              str[nbBits-i-1] = '1';
49          else
50              str[nbBits-i-1] = '0';
51      }
52
53      return str;
54  }
55
56  int main(int argc, char* argv []) {
57      number input;
58      char* str;
59
60      if(argc < 2) {
61          fprintf(stderr, "Usage: convert entier\n\n");
62          return -1;
63      }
64
65      if(convert(&input, argv[1]) != 0) {
66          fprintf(stderr, "%s", errorMsg);
67          return -1;
68      }
69
70      if( (str = getStr(input)) == NULL) {
71          fprintf(stderr, "%s", errorMsg);
72          return -1;
73      }
74
75      printf("%s\n", str);
76
77      free(str);
78
79      return 0;
80  }
```

makefile

```
1  all: convert
2
3  convert: convert.c
4          gcc convert.c -o convert -lm
```