IUT de Montpellier **21 décembre 2017**
Département Informatique **Durée : 3h00**
2ème année

SYSTEME et RESEAUX

AVERTISSEMENT : Ceci est un sujet commun pour les matières : système et réseaux.
Vous devrez, cependant, rédiger sur deux copies différentes :
- une copie pour la partie réseau (1ère Partie – Questions 1,2,3,4)
- une copie pour la partie système (2ème Partie – Questions 1,2,3,4,5,6,7)

1 - Présentation du contexte

Le centre commercial "Les portes de Montpellier", souhaite mettre en place un système de mesures modernes pour mieux gérer les flux de clients.

Pour cela il va mettre en place une multitude d'objets connectés, qui seront des capteurs pour compter le nombre de passages. Ces objets seront répartis en 4 groupes (cf Annexe 1 - Plan du centre) :

- Le premier groupe , est un ensemble de capteurs, disposés aux 3 entrées des parkings , qui comptera les véhicules qui entrent et sortent.
- Le deuxième groupe, disposés aux 3 entrées du Centre, comptera les entrées et sorties de personnes dans les locaux du centre commercial.
- Le troisième groupe, disposé aux deux entrées du magasin comptera les personnes qui entrent.
- Le quatrième, disposés aux caisses, permettra de compter les passages.

Ces objets qui utilisent la technologie (IoT) sont de simples capteurs lumineux qui incrémentent un compteur dès qu'il y a un passage. Leur particularités, c'est qu'ils utilisent un réseau de transmission sans fil de faible débit (LoRa).

2 - Fonctionnement général

Le fonctionnement général de cette installation est décrite ci-dessous et schématisé en ANNEXE 2.

Possédant de faibles capacités de stockage , un objet transmet un message à une machine située à l'Accueil Client du centre commercial toutes les 5 minutes. La structure des données transmise est décrite dans l'annexe 3.

La machine de l'accueil est en charge de mettre en forme les données pour les transmettre à une base de données qui centralise toutes les mesures . En fait cette machine totalise par type de capteur le nombre total de mouvements enregistrés et envoie toutes les 10 minutes un message ayant la structure ci-dessous :

15:12:2017/10:10/Park:IN:20/Park:OUT:01/Center:IN:50/Center:OUT:30/Entree:IN:30/Caisse:OUT:30
Interprétation : Le 15/12/2017 à 10h10 entre 10h et 10h10 il a été observé :
20 voitures qui entrent et 1 qui sortent dans le parking
50 personnes qui entrent et 30 qui sortent du centre commercial,
...

Cette base se situe sur un serveur, dans le bâtiment administratif, situé à quelques dizaines de mètres du centre commercial.

1ère Partie – Réseaux

1 - Concevoir le schéma du réseau de cet établissement. Vous préciserez les équipements nécessaires et leur type, comment ils sont connectés, ainsi que les types de supports (câbles) à utiliser.

2 - Dans ce type de réseau, les problèmes de sécurité s'articulent essentiellement autour de la disponibilité du système. En effet , les objets transmettant exclusivement des valeurs correspondant à un comptage, leur interception ne pose pas de problème. Par contre , si un ou plusieurs composants (machines serveur incluses) venaient à ne plus fonctionner, cela va impacter l'organisation de l'établissement.

- Quelles peuvent être les causes de blocage de ce système ?
- Quelles mesures de protection peut-on envisager ?

3 - Sachant que l'établissement possède l'adresse réseau : 172.38.0.0, proposez un plan d'adressage de ce réseau, en prenant en compte les objectifs de sécurité de la question précédente.

4 - Les objets communiquant via le protocole TCP/IP , ils envoient les données à une application (serveur) située sur machine de l'accueil. Cette application serveur est accessible à l'adresse 10.20.1.10 et le port 1111. La logique d'échange est simple, le capteur se connecte au serveur, il envoie sa mesure, le serveur renvoie un accusé de réception : la chaine "ACK", si la valeur est bien reçue et se déconnecte.

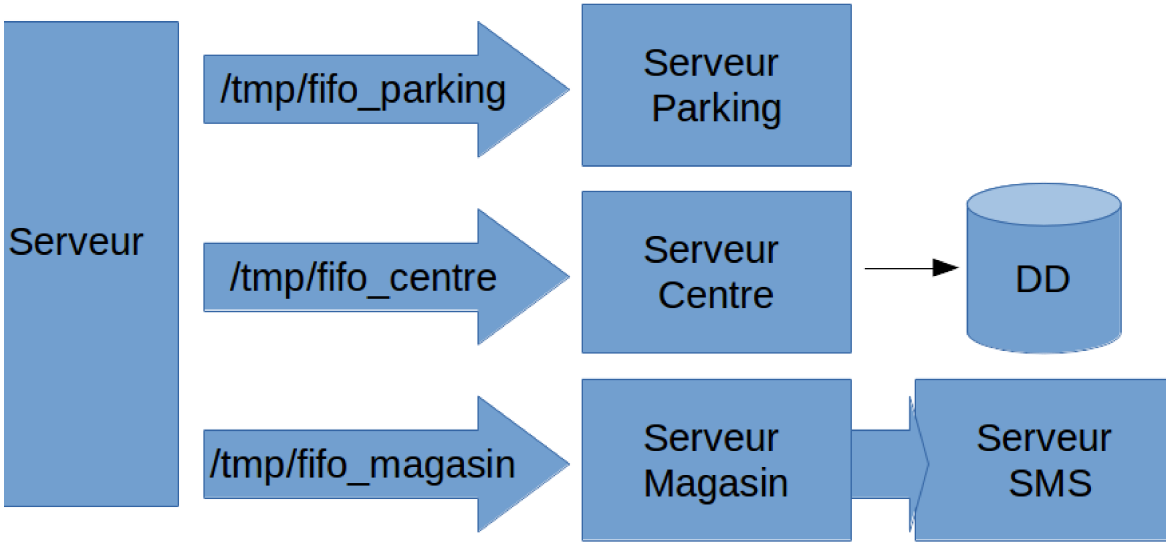
S'il y a eu un problème le serveur retourne "NACK". Dans ce cas le capteur retransmet la mesure. En cas de nouvelle erreur il réessaye 2 fois, puis se déconnecte.

En utilisant les signatures des fonctions de l'annexe 4, proposer le code (ou pseudo-code) de l'application cliente, située sur les capteurs. Des explications dans ce code seront très appréciées du correcteur.

Remarque : Pour simplifier l'application , on met à votre disposition une fonction : **set_addr(Addr_IP4, Port)** qui va mettre en forme les différents champs de **sock-addr**.

2ème Partie – Système

Dans cette partie nous allons réaliser les serveurs qui vont traiter les messages reçus par les capteurs. Le système général est décrit dans le schéma ci-dessous:



Remarque : Avant de commencer les questions, il est impératif de consulter le fichier «centre.h » en annexe 5 qui décrit les différentes structures utilisées dans la suite.

Préambule : Gestion d'un journal par une liste

Nous souhaitons garder un journal complet des entrées/sorties concernant les véhicules et personnes afin de réaliser un bilan à n'importe quel moment. Pour cela nous allons gérer trois listes qui vont représenter : (i) les véhicules dans le parking ; (2) les personnes dans le centre ; (3) et les personnes dans le magasin. Ces trois listes sont déclarées dans le fichier centre.h comme suit :

```
extern node_t* personne_centre_list;
extern node_t* personne_magasin_list;
extern node_t* voiture_parking_list;
```

Question 1: Enregistrement des transactions de flux

Nous souhaitons écrire une fonction générique qui permet d'insérer en tête de liste un nouveau flux. Voici la signature de la fonction:

```
node_t* insertion_nouvelle_transaction(node_t* tete, int in, int out);
```

tete représente la tête actuelle de la liste; *in* et *out* sont des nombres positifs qui représentent respectivement le nombre d'entrées et le nombre de sorties pour cette transaction ; et la fonction retourne une nouvelle tête pour la liste.

Donnez le code de la fonction *insertion_nouvelle_transaction*. Cette fonction devra allouer dynamiquement la mémoire pour une structure *node_t*, remplir correctement les différents champs et retourner cette adresse comme nouvelle tête de liste.

Réception des commandes et routage vers les serveurs de traitement

Le serveur d'application définit une fonction principale *traiter_donnees(char *donnees)* qui va recevoir un message au format présenté dans la partie 1 (réseau) et qu'il doit transmettre aux serveurs de traitement.

Voici le code de la fonction *traiter_donnees(char* donnees)* :

```
void traiter_donnees(char *donnees) {
traiter_donnees_parking(donnees);
traiter_donnees_centre(donnees);
traiter_donnees_magasin(donnees);
}
```

Notre objectif est de réaliser ces trois fonctions ainsi que le code des serveurs de traitement.

Pour rappel, les données sont transmises via le réseau sous la forme d'une chaîne de caractères construite avec des champs à taille fixe (les nombres auront toujours 2 chiffres) comme le montre l'exemple ci-dessous :

Date (10)	Heure (5)	Parking (10)	Parking (11)	Centre (12)	Centre (13)	Magasin (12)	Magasin (13)
15:12:2017	**10:10**	**Park:IN:20**	**Park:OUT:01**	**Center:IN:50**	**Center:OUT:30**	**Entree:IN:30**	**Caisse:OUT:30**

Nous proposons le code suivant pour la fonction *traiter_donnees_parking*.

```
#include "centre.h"
void traiter_donnees_parking(char* donnees) {
  int annee, mois, jour, heure, min, in, out;
  donnee_vehicule_t enregistrement ; // voir centre.h
  recuperer_les_champs_utils(donnees,&annee,&mois,&jour,&heure,&min,&in,&out);
  enregistrement.date.annee = annee;
  enregistrement.date.mois = mois;
  enregistrement.date.jour = jour;
  enregistrement.date.heure = heure;
  enregistrement.date.minute = min;
  enregistrement.in = in;
  enregistrement.out = out;
  envoyerAuServeurParking(&enregistrement);
}
```

Question 2: Récupération des champs

Écrivez le code de la fonction:

```
recuperer_les_champs_utils(char* msg,int* y,int* m,int* d, int* h, int* min, int* i, int* o);
```

qui va récupérer les différents champs du message et décoder leurs valeurs en entier (integer) afin de les stocker dans les variables passées comme paramètres.

Astuce : Vous pouvez utiliser les fonctions atoi et memcpy décrites en annexe et utiliser un tableau de caractères intermédiaire de taille suffisante pour copier les champs et les traiter ensuite. Rappel important: en C une chaîne de caractère se termine toujours par '\0'.

Question 3: Gestion du Parking

Compléter le code ci-dessous de la fonction *envoyerAuServeurParking* pour envoyer les informations en utilisant le tube nommé :

```
#include "centre.h" // a consulter en annexe
void envoyerAuServeurParking(donnee_vehicule_t* data_ptr  ){
int descripteur ; //descripteur pour la fifo
//TODO ouverture du pipe nomme avec le bon mode
```

```
//TODO Ecriture des donnees

//TODO fermeture du descripteur

}
```

Question 4: Serveur parking

Complétez le code du serveur « serveur_parking.c » donné ci dessous:

```
#include "centre.h" // a consulter en annexe 5
//liste chainee des transactions recues pour les couples (entrees, sorties)
node_t* voiture_parking_list = NULL ;
int main(int argc, char** argv){
  int sortir = 0 ; //variable pour indiquer la fin du traitement
  int descripteurEntree ; //descripteur pour la lecture des donnees
  donnee_vehicule_t donneeTampon;
  voiture_parking_list = NULL;
  //TODO:creation du tube nomme pour recevoir les donnees
  //...
  /// Une boucle principale de serveur
  while(1) {
    //TODO: Ouverture de la fifo
    descripteurEntree = ???;
    if ( descripteurEntree == -1 ) {
      fprintf(stderr, "erreur d ouverture du fichier %s\n",PATH_FIFO_PARKING);
      exit(1);
    }
    // Une boucle de traitement pour recevoir les donnees par le tube
    sortir = 0;
    while(!sortir) {
      //TODO: Lecture d'un message de la fifo dans la variable donneeTampon
      int n = ???;
      if(n>0) {
      //TODO: allocation d un nouveau noeud de liste
      node_t* nouveau_noeud = ...;
      //TODO recuperation des donnees utiles

      //TODO inserer le nouveau dans la liste et mettre a jour la tete de liste
      } else {
      sortir = 1;
      }
    }
    // Fermeture du descripteur de la fifo
    close(descripteurEntree);
  }
}
```

Gestion des personnes au centre commercial

Nous souhaitons gérer les entrées et sorties des personnes au centre commercial. Pour cela nous devons traiter les messages afin d'enregistrer les flux dans la liste *personne_centre_list*
Pour des raisons de sécurité, les transactions devront être enregistrées dans un fichier texte ligne par ligne avec le format suivant « in out \n» où *in* et *out* sont des entiers qui représentent respectivement les entrées et sorties.
Par exemple le fichier suivant :

10 0
5 6
0 9
indique trois transactions avec comme entrées 10, 5 et 0 et comme sorties 0, 6 et 9.

Question 5 :

Donnez le code du serveur_centre.c avec la gestion de la sauvegarde dans un fichier (le chemin du fichier se trouve dans centre.h)

Gestion du magasin et les alertes

Nous souhaitons à présent gérer les entrées et sorties des personnes dans le magasin afin d'analyser le trafic et de pouvoir envoyer des alertes selon le taux de remplissage. Les flux du magasin sont enregistrés dans la liste `personne magasin list.`

Question 6: Bilan

Afin de connaître le nombre exact des personnes présentes dans le magasin , donnez le code de la fonction ***`int bilan(node_t* tete);`*** qui doit comptabiliser le nombre de personnes encore présentes dans le magasin.

Pour gérer les différentes situations d'alertes trois programmes *alerte80*, *alerte90* et *alerte95* ont été développés et vous sont donc donnés. Ces programmes lisent sur leur entrée standard (stdin) un numéro de téléphone et se charge de d'envoyer un sms afin de prévenir que les taux de remplissage de 80 %, 90 % et 95 % ont été franchis.
Voici un exemple d'utilisation si le taux d'occupation du magasin vient de dépasser 90 % :
$ echo '+33690909090' | alerte90

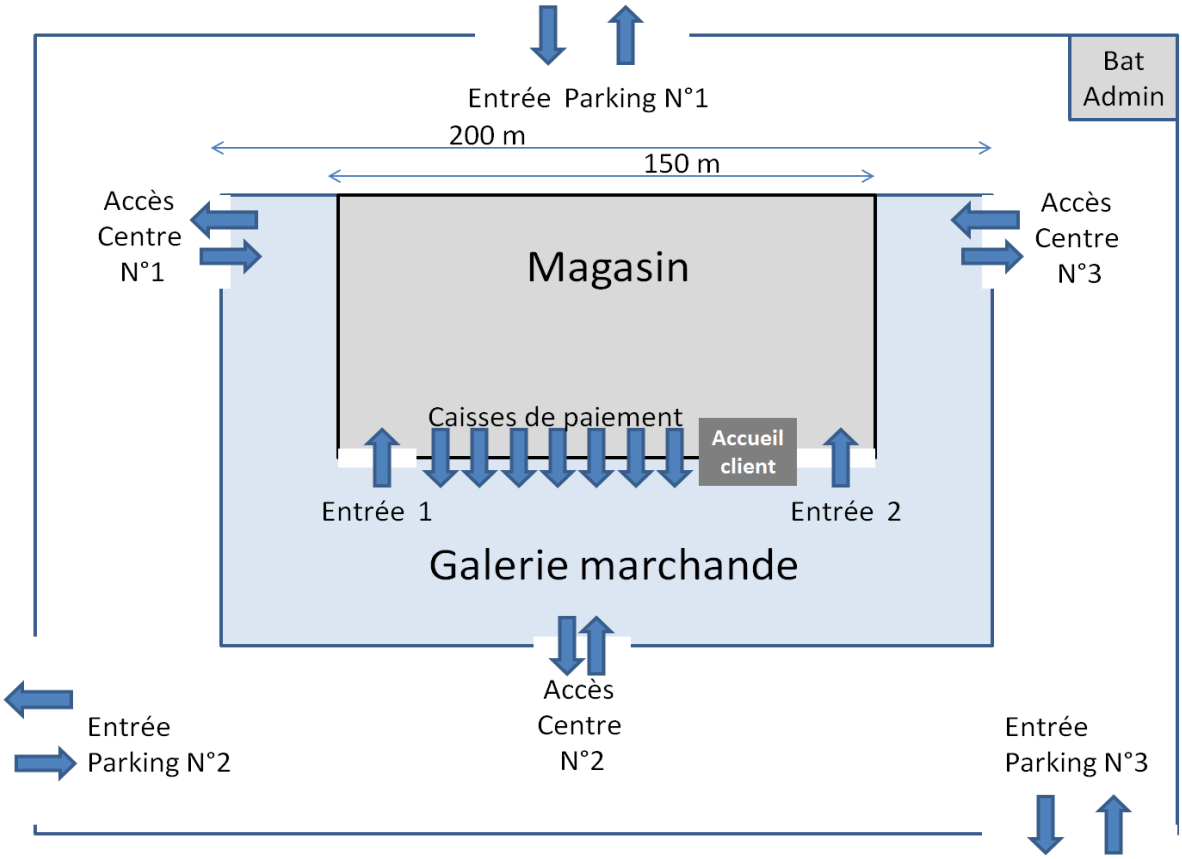
Le serveur du magasin va donc potentiellement lancer l'un des trois programmes d'alerte en lui fournissant le numéro de téléphone, défini par la macro PHONE, dans son entrée standard. Le taux de remplissage du magasin est calculé en divisant le nombre de personnes encore dans le magasin par la constante NB_MAX définie dans 'centre.h'.

Question 7 :

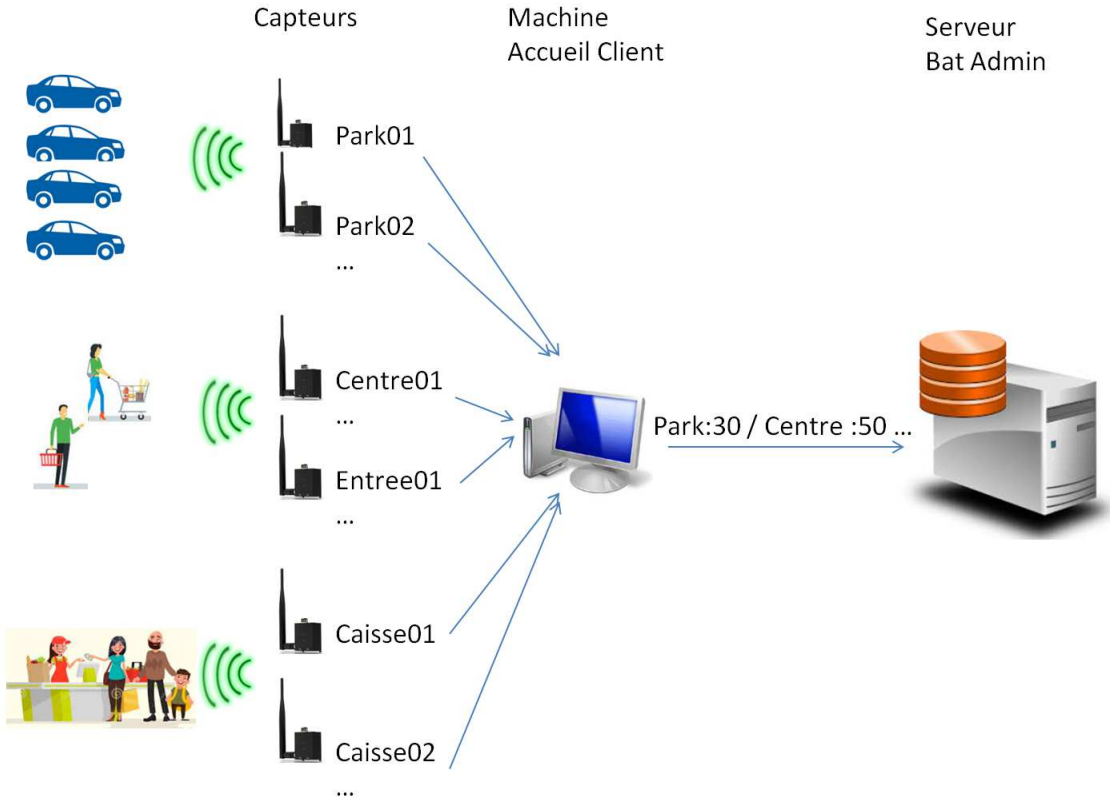
Donnez le code du serveur_magasin.c avec la gestion des alertes.

Annexes

ANNEXE 1 - Plan du centre



ANNEXE 2 - Schéma général de fonctionnement



ANNEXE 3 - Structure des données venant d'un capteur

Une mesure envoyée par un objet est une chaine de caractères qui possède le format ASCII suivant :

PARK01/15:12:2017/10:05/OUTPUT/10 --> 10 voitures ont quitté le parking entre 10h et 10h05 par l'entrée n°1

SHOP02/15:12:2017/10:10/INPUT/35 --> 35 personnes sont entrés dans le magasin entre 10h05 et 10h10 par l'entrée n° 2

ANNEXE 4 - Fonctions C

```
int socket(famille,type,0);
bind(int descripteur,sockaddr localaddr,int addrlen);
int listen(int socket,int backlog);
int accept(int socket,struct sockaddr * addr,int * addrlen);
int connect(int socket,struct sockaddr * addr,int * addrlen);
int recv(int socket,char * buffer,int len,0);
int send(int socket,char * buffer,int len,0);
int close(int socket);
```

ANNEXE 5 : centre.h

```
#ifndef CENTRE_H
#define CENTRE_H

typedef struct date_t {
  int annee;
  int mois;
  int jour;
  int heure;
  int minute;
} date_t;

typedef struct donnee_vehicule_t {
  date_t date;
  int in;
  int out;
} donnee_vehicule_t;

#define CENTRE 1
#define MAGASIN 2

typedef struct donnee_personne_t {
  date_t date;
  int localisation; //soit CENTRE ou MAGASIN
  int in;
  int out;
} donnee_personne_t;

// -- des constantes pour nommer les fifos
#define PATH_FIFO_PARKING "/tmp/fifo_parking"
#define PATH_FIFO_CENTRE "/tmp/fifo_centre"
#define PATH_FIFO_MAGASIN "/tmp/fifo_magasin"

//--nom de fichier sauvegarde du centre
#define STOCKAGE_PERSONNE_CENTRE "/data/personne.data"
```

```
//--- Gestion des listes
typedef struct node_t {
  struct node_t* suivant;
  int in;
  int out;
} node_t;

node_t* insertion_nouvelle_transaction  (node_t* tete, int in, int out);

//donne le bilan d'une liste
int bilan(node_t* tete);

//nb max de personne dans le magasin
#define NB_MAX 1000
#define PHONE "+33690909090"

//
extern node_t* personne_centre_list;
extern node_t* personne_magasin_list;
extern node_t* voiture_parking_list;

#endif
```

ANNEXE 6 - Manuel des fonctions C

int atoi(const char *_nptr_); La fonction atoi() convertit le début de la chaîne pointée par *nptr* en entier de type *int.*

void *memcpy(void *_dest_, const void *_src_, size_t _n_); La fonction memcpy() copie *n* octets depuis la zone mémoire *src* vers la zone mémoire *dest*.

int open(const char *_pathname_, int _flags_);

Ouverture d'un fichier indiqué par le pathname. Le paramètre *flags* doit inclure l'un des *mode d'accès* suivants : O_RDONLY, O_WRONLY ou O_RDWR.

ssize_t read(int _fd_, void *_buf_, size_t _count_);

read() lit jusqu'à *count* octets depuis le descripteur de fichier *fd* dans le tampon pointé par *buf.*

ssize_t write(int _fd_, const void *_buf_, size_t _count_);

write() écrit jusqu'à *count* octets dans le fichier associé au descripteur *fd* depuis le tampon pointé par *buf*.

int close(int _fd_); ferme de descripteur fd.

FILE *popen(const char *_commande_, const char *_type_);

La fonction popen() engendre un processus en créant un tube. L'argument *commande* est un pointeur sur une chaîne de caractères. L'argument *type* est un pointeur sur une chaîne de caractères qui doit contenir « r » pour la lecture ou « w » pour l'écriture.

int pclose(FILE *_stream_); Pclose() ferme le flux ouvert par popen().

int mkfifo(const char *_pathname_, mode_t _mode_);
La fonction **mkfifo**() crée un fichier spécial FIFO (tube nommé) à l'emplacement *pathname*. *mode* indique les permissions d'accès. La valeur renvoyée par **mkfifo**() est 0 si elle réussit, ou -1 si elle échoue

int sprintf(char *_str_, const char *_format_, ...);

La fonction sprintf effectue une écriture formatée dans la chaîne str. Elle renvoie une valeur négative en cas d'erreur.

size_t strlen(const char *_s_);

Cette fonction renvoie la taille de la chaîne de caractères s.