TD 6

semaine du 8/5/00

Objectifs : la ligne de commande et les variables d'environnement

Exercice 1 : Affichage des variables d'environnement d'un programme C

Lorsqu'on exécute un programme, on peut passer des arguments et les variables d'environnement sur la ligne de commande. En langage C, ces arguments peuvent être récupérés par la fonction main() du programme. Le prototype de la fonction main est dans ce cas :

void main (int argc, char *argv[], char *arge[]) ;

/** argc : nombre d'arguments de la ligne de commande

argv : tableau de pointeurs de chaînes de caractères contenant les différents arguments, la première chaîne contient le nom de l'exécutable.

arge : tableau de pointeurs de chaînes de caractères contenant les valeurs des différentes variables d'environnement, la dernière chaîne est à NULL.

*/

Ecrire un programme en C qui affiche les valeurs des variables d'environnement.

Exercice 2 : Exemple d'utilisation des variables d'environnement

Le programme suivant (en C) affiche tous les rendez-vous d'une journée dont la date est passée en paramètre. On peut appeler par exemple

> rendezvous 20:04:2000

La recherche des chaînes de caractères, etc., est fait par la fonction search(char* date, char* nomDeFichier) qui vous est fournie dans le fichier rdvSearch.c. Il vous est seulement demandé de compléter le fichier rendezvous.c en initialisant la variable datafile (qui contient le nom du fichier dans lequel figurent les rendez-vous). Pour comprendre le code de search() (ce qui n'est pas indispensable) : les rendez-vous sont séparés par une ligne blanche, la date est en tête.

2.1 Version 1 : le fichier dans lequel les rendez-vous figurent est fixé par la variable d'environnement RENDEZVOUS. Si ils sont par exemple dans ~/prive/monAgenda, il faut lancer au loggin la commande setenv RENDEZVOUS ~/prive/monAgenda. Si la variable d'environnement RENDEZVOUS n'est pas définie, le programme affiche une erreur. Vous pouvez utiliser la fonction de bibliothèque getenv().

```
#define MAINFILE
#include "environ2.h"


 /* getenv() renvoie un pointeur sur la valeur (chaîne de caractère) de la variable d'environnement name
    dans l'environnement, NULL si la variable n'est pas définie. */
char* getenv(char *name);


int main(int argc , char *argv[] , char *arge[]){
char * sptr, datafile[128] ;
  int datepos ;

   /* Recherche de la variable RENDEZVOUS dans l'environnement */
        /*                à compléter                */
```

```
/* Si la variable d'environnement  RENDEZVOUS est définie, copie de sa valeur dans datafile[] */
                                /*                à compléter                */
```

```
/* Si la variable d'environnement  RENDEZVOUS n'est pas définie message d'erreur et sortie.
a modifier dans les versions 2 et 3 */
                                        /*              à compléter             */


/*  Lecture du motif à rechercher, à modifier dans la version 3*/
    if (argc == 2)
          datepos = 1 ;                                 /*En version 1 et 2, la date est dans argv[1] */
      else
          {printf("Usage %s jj:mm:aaaa\n",argv[0]); /* modifier en version 3*/
          exit(2);}                                             /* c'est la 2eme fin en erreur */

/* Lancement de la recherche */
    search(argv[datepos], datafile) ;
}
```

2.2 Version 2 : on suppose maintenant que le programme prend un fichier par défaut dans le répertoire de connexion (fichier caché ${HOME}/`.rendezvous`) si la variable RENDEZVOUS n'est pas définie. Cette possibilité est moins prioritaire que la variable d'environnement. Modifiez le programme là où c'est indiqué (si le fichier ${home}/.rendezvous n'existe pas, on laissera le système traiter l'erreur).

2.3 Version 3 : on veut aussi pouvoir passer le fichier où sont notés les rendez vous en paramètre d'appel de la fonction, cette dernière possibilité étant la plus prioritaire. On peut donc appeler :

> rendezvous 20:04:2000 ou

> rendezvous –f ~/travail/agendabis 20:04:2000

[Un argument commençant par un signe moins (ici –f) est la convention des programmes sous Unix pour introduire une option (un paramètre facultatif).]

Modifier le programme pour ajouter cette possibilité. L'emplacement des modifications est indiqué.

Rappel : Pour chercher une chaine dans une autre :

strstr(char *s1, char *s2) trouve la première occurence de la chaîne s2 (à l'exclusion du caractère terminal '\0') dans la chaîne s1. strstr() renvoie un pointeur sur la chaîne trouvée, ou un pointeur nul dans le cas où aucune chaîne n'a été trouvée. Quand s2 pointe sur une chaîne de longueur nulle (autrement dit sur la chaine "") strstr() renvoie s1.

strcpy(char *s1, char *s2) copie la chaïne s2 à l'adresse s1 (y compris le caractère final '\0').

strcat(char *s1,char *s2) concatène la chaîne s2 à la suite de la chaîne à l'adresse s1 (le premier caractère de s2 remplace le '\0' final de s1)