DAEMONLAB

EXAMEN

RED ALERT

Programmation Système sur Unix

- DaemonLab -

INDEX

01 – Avant-propos

Votre travail doit être rendu via le dossier ~/exam/ra_allied/ dans votre espace personnel.

SI VOUS FAITES ERREUR ET QUE LE DOSSIER QUE VOUS UTILISEZ POUR VOTRE RENDU EST DIFFERENT, VOUS NE SEREZ PAS EVALUE FAUTE D'AVOIR PU TROUVER VOTRE TRAVAIL.

Ce travail est a effectuer seul. De plus il s'agit d'un examen. Vous n'avez donc pas le droit de communiquer avec vos camarades.

Votre rendu doit respecter strictement l'ensemble des règles suivantes :

- Il ne doit contenir aucun fichier objet. (*.o)
- Il ne doit contenir aucun fichier tampon. (*~, #*#)
- Il ne doit pas contenir votre production finale (programme ou bibliothèque)

LA PRESENCE D'UN FICHIER INTERDIT METTRA IMMEDIATEMENT FIN A VOTRE EVALUATION !

Votre programme doit respecter les Tables de la Norme dans leur intégralité. Vous êtes invité à les observer depuis l'Infosphère. Elles sont disponibles comme ressource de cette activité.

02 – Fonctions autorisées

La bibliothèque logicielle venant avec le C est vaste et disponible. La LibLapin, que vous utilisez dans vos projets multimédia, est également vaste… Cependant nous avons fait le choix de vous interdire leurs utilisation intégrales, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

L'UTILISATION D'UNE FONCTION INTERDITE EST ASSIMILEE A DE LA TRICHE. LA TRICHE PROVOQUE L'ARRET DE L'EVALUATION ET LA PERTE DES MEDAILLES.

Vous n'avez le droit d'utiliser aucune fonction issue de la LibC ou de la LibLapin à l'exception de celles que nous vous autoriserons explicitement.

POUVOIR UTILISER UNE FONCTION NE SIGNIFIE PAS NECESSAIREMENT QUE CELLE-CI SOIT UTILE A VOTRE CAS.

Pour cette activité, issu de la LibC, vous n'avez le droit qu'à la liste suivante :

- open
- close
- read
- srand
- cos
- atan2
- time
- opendir
- readdir
- write
- alloca
- atexit
- rand
- sin
- sqrt
- malloc, free
- closedir
- stat

03 – Lire sur l'entrée standard

Fichier read_data.c

« LACHER VOS HAMBURGERS ! LES ROUGES VONT BIENTOT TRANSMETTRE ! PREPAREZ LE RECEPTEUR, INTERCEPTEZ LA TRANSMISSION ET RENDEZ FIERE VOS MERES ! CETTE COMMUNICATION EST IMPORTANTE POUR LES COMMIS. LE PENTAGONE SOUHAITE EN APPRENDRE LE PLUS POSSIBLE. TERMINE ! »

Implémentez la fonction suivante :

```
int       std_read_data(int             fd,
                        char            *str,
                        size_t          len)
```

Cette fonction lit dans **fd** tant qu'elle n'a pas lu **len - 1** octets et placé ceux-là dans **str**. Le dernier octet doit être '\0'. Si **read** renvoi 0 ou -1, la fonctions s'arrête.

La fonction renverra le nombre d'octets lu ou -1 si une erreur est rencontrée.

04 – Sauvegarder dans un fichier

Fichier write_file.c

« VOUS AVEZ FAIT RECULER LE COMMUNISME, RECRUE ! VOUS FAITES HONNEUR AUX DOLLARS INVESTIS DANS VOTRE FORMATION. NE PERDEZ AUCUNE DES INFORMATIONS QUE VOUS AVEZ RECUPERE ! ENTREZ TOUTES LES INFORMATIONS DANS L'IBM ! FAITES COULER DE LA LIBERTE DANS SES CIRCUITS ! TERMINE !»

Implémentez la fonction **std_write_file** :

```
int             std_write_file(const char     *filename,
                               char           *data,
                               size_t         len)
```

Cette fonction écrit dans **filename** tant qu'elle n'a pas écrit **len** octets issus de **data**. Si **write** renvoi 0 ou -1, la fonction s'arrête. Attention, write ne va pas forcement tout écrire d'un coup ! Vous devez donc vous assurer de l'écriture des **len** octets à l'aide d'une boucle.

La fonction renverra le nombre d'octets écrits ou -1 si une erreur est rencontrée.

05 – Parcourir un dossier

Fichier browse_directory.c

« L'ENREGISTREMENT DES DONNEES S'EST BIEN PASSE ! LE PENTAGONE NOUS ANNONCE LA DECOUVERTE D'UNE FAILLE DANS LE SYSTEME INFORMATIQUE SOVIETIQUE ! VOUS ALLEZ L'EXPLOITER AVANT QUE CES VERMINES NE DECOUVRENT QUE NOUS AVONS CORROMPU LEUR CAMARADE DORIS ! FAITES AU PLUS SIMPLE, LISTEZ LES ENTREES SITUE A LA BASE DU SYSTEME. TERMINE ! »

Implémentez la fonction suivante :

```
typedef struct      s_file
{
     char           name[128];
     int            size;
}                   t_file;

int       std_browse_directory(const char    *dir,
                               t_file        *out,
                               size_t        len)
```

Cette fonction ouvre le dossier passé dans **dir** et en parcoure le contenu.

Les fichiers (et non les dossiers) doivent être renseigné dans **out** jusqu'à un maximum de **len.** Si il y a plus de fichiers que d'espace dans **out,** les éléments surnuméraires ne sont pas stockés. La fonction renvoi le nombre de fichiers trouvés (quelque soit **len**).

Le remplissage du tableau de structure **out** se fait par le remplissage des deux champs de la structure **t_file**.