

PROJET SYSTÈME D' EXPLOITATION / UNIX

SUJET 1

FORUMS

Il s'agit de concevoir un gestionnaire de forums local à une machine.

La gestion des forums est assurée par un serveur lancé une seule fois à l'initialisation du système (démon), en arrière-plan par la commande **servforum** :

syntaxe : servforum

Chaque utilisateur a la possibilité de demander :

- la création d'un forum sur un nouveau thème
- son abonnement à un forum
- de poster un message dans un forum
- de consulter les messages non lus d'un forum auquel il est abonné
- de le désabonner
- ...

De plus le serveur doit détruire tous les messages trop anciens.

On demande également de conserver une trace de toutes les opérations effectuées par le serveur.

P.S. : dans la mesure du possible vous ferez afficher des informations sur les ressources utilisées par vos programmes à la fin de leur exécution.

PROJET SYSTÈME D' EXPLOITATION / UNIX

SUJET 2

SURVEILLANCE DE L' EVOLUTION D' UNE ARBORESCENCE

Il s'agit de fournir un moyen d'observer l'évolution des fichiers d'une arborescence donnée, à intervalle de temps régulier.

A l'instant t de l'observation on déterminera pour chaque fichier si par rapport à la dernière observation (à l'instant t-1) il est :

Nouveau
Modifié taille, date de dernière modification
Inchangé
Supprimé

La mise en œuvre d'une telle application nécessite l'écriture d'une commande **analysarbo** lancé en arrière plan (démon) qui analyse périodiquement l'état de l'arborescence :

syntaxe : **analysarbo** -i <périodicité> -d <rep> -f <fic>

avec :

périodicité : prenant ses valeurs dans {jour, semaine, mois}

rep : nom du répertoire racine de l'arborescence à observer

fic : nom du fichier qui contient l'état de l'arborescence lors de la dernière observation

Pour connaître l'état de l'arborescence l'utilisateur lancera la commande **etatarbo** qui exploitera le fichier fic,

syntaxe : **etatarbo** [-cr] [-ls] [-lm] [-ln] [-li] [-ld] [-co] [-ec]

avec les options :

- cr : pour obtenir un compte-rendu synthétique de l'analyse de l'arborescence (nombre total de fichiers, nombre de fichiers supprimés, de fichiers nouveaux, ...) à partir de la dernière analyse
- ls : pour la liste des fichiers supprimés
- lm : " modifiés
- ln : " nouveaux
- li : " inchangés
- ld : " doublons (ayant le même nom relatif).
Pour chaque doublon, on précisera si les fichiers ont des contenus identiques ou différents
- ec : pour obtenir l'état complet de l'arborescence

Prévoir une version de votre application qui pourra analyser l'arborescence de façon ponctuelle (non périodique) afin par exemple de se rendre compte immédiatement de l'effet de l'installation d'une nouvelle application.

PROJET SYSTÈME D'EXPLOITATION / UNIX

SUJET 3

SPOOLER d'IMPRESSIONS

Écrire un gestionnaire de demandes d'impression en respectant les priorités attachées aux demandes ; ces priorités sont fonction du groupe d'appartenance de l'utilisateur qui lance la commande d'impression. Les groupes d'utilisateurs sont par priorité décroissante :

- iup3
- iup2
- iup1
- dessacsi
- desssid
- profs

Les demandes proviennent d'utilisateurs différents, mais travaillant tous sur la même station et sont formulées par la commande "**imprime**" :

syntaxe : `imprime [-n nb] [-t "titre"] <nom relatif du fichier>`

options :
-n : imprimer nb exemplaires
-t "titre" : ajoute le texte "titre" dans l'entête

Le gestionnaire sera lancé une seule fois à l'initialisation du système, en arrière plan par la commande "**gestimp**" (démon) :

syntaxe : `gestimp`

L'impression d'un fichier sera précédée d'une en-tête comportant le nom du fichier, le nom du propriétaire, la taille en octets, la date de dernière modification du fichier, l'heure de demande d'impression et éventuellement d'un titre (voir l'option -t).

La commande "**infoimp**" permettra d'afficher la liste des demandes en attente ; et la commande "**annule**" permettra d'annuler une demande d'impression.

On vous demande également d'étudier les problèmes de droits d'impression et d'annulation.

Nous vous conseillons de réaliser ce projet à l'aide des files de messages, outils de communication inter processus d'UNIX. Pour des raisons pratiques les impressions seront en réalité des afficha-ges à l'écran effectués par la commande `cat`.

P.S. : dans la mesure du possible vous ferez afficher des informations sur les ressources utilisées par vos programmes à la fin de leur exécution.

PROJET SYSTÈME D' EXPLOITATION / UNIX

SUJET 4

SIMULATION d'APPLICATION TEMPS RÉEL

Solution Thread

Nous nous plaçons dans un environnement industriel. Il s'agit de visualiser l'évolution du remplissage d'un entrepôt en temps réel. Cet entrepôt est « alimenté » par deux machines d'usinage.

On dispose d'une première machine d'usinage pouvant traiter des pièces de type A et de type B et d'une seconde machine traitant des pièces de type B et type C.

Chaque machine reçoit aléatoirement des pièces de type A, B, ou C et renvoie à l'autre machine les pièces qu'elle ne peut pas traiter.

Ces deux machines fonctionnent en parallèle.

Quand une machine a fini de traiter une pièce elle l'envoie à un "robot" en lui indiquant l'endroit où il faut la ranger dans l'entrepôt, sachant que les pièces de même type sont regroupées au même endroit.

Pour simuler cette application nous vous conseillons de la découper en quatre threads lancées à partir d'un unique programme et qui s'exécuteront en parallèle.

1^{ère} tâche : génération des pièces brutes de type A, B, C qu'elle envoie à tour de rôle aux deux machines d'usinage (cette tâche ne connaît pas la "fonction" de chaque machine)

2^{ème} tâche : simulation de la première machine d'usinage

3^{ème} tâche : simulation de la deuxième machine d'usinage

4^{ème} tâche : visualisation de l'évolution de l'occupation de l'entrepôt

Le lancement de l'application se fera par la commande **entrepot**.

syntaxe : **entrepot** [-n <nombre de pièces>]

(N.B. : l'entrepôt est de taille 2*nombre de pièces envoyées à chaque machine)

Pour une démonstration qui illustre bien la simulation on affectera des vitesses de fabrication différentes selon les différentes machines et les différentes pièces.

P.S. : dans la mesure du possible vous ferez afficher des informations sur les ressources utilisées par vos programmes à la fin de leur exécution.

PROJET SYSTÈME D' EXPLOITATION / UNIX

SUJET 5

SIMULATION d'APPLICATION TEMPS RÉEL

Solution Processus

Nous nous plaçons dans un environnement industriel. Il s'agit de visualiser l'évolution du remplissage d'un entrepôt en temps réel. Cet entrepôt est « alimenté » par deux machines d'usinage.

On dispose d'une première machine d'usinage pouvant traiter des pièces de type A et de type B et d'une seconde machine traitant des pièces de type B et type C.

Chaque machine reçoit aléatoirement des pièces de type A, B, ou C et renvoie à l'autre machine les pièces qu'elle ne peut pas traiter.

Ces deux machines fonctionnent en parallèle.

Quand une machine a fini de traiter une pièce elle l'envoie à un "robot" en lui indiquant l'endroit où il faut la ranger dans l'entrepôt, sachant que les pièces de même type sont regroupées au même endroit.

Pour simuler cette application nous vous conseillons de la découper en quatre processus lancés à partir d'un unique programme et qui s'exécuteront en parallèle.

1^{ère} tâche : génération des pièces brutes de type A, B, C qu'elle envoie à tour de rôle aux deux machines d'usinage (cette tâche ne connaît pas la "fonction" de chaque machine)

2^{ème} tâche : simulation de la première machine d'usinage

3^{ème} tâche : simulation de la deuxième machine d'usinage

4^{ème} tâche : visualisation de l'évolution de l'occupation de l'entrepôt

Le lancement de l'application se fera par la commande **entrepot**.

syntaxe : **entrepot** [-n <nombre de pièces>]

(N.B. : l'entrepôt est de taille 2*nombre de pièces envoyées à chaque machine)

Pour une démonstration qui illustre bien la simulation on affectera des vitesses de fabrication différentes selon les différentes machines et les différentes pièces.

P.S. : dans la mesure du possible vous ferez afficher des informations sur les ressources utilisées par vos programmes à la fin de leur exécution.

PROJET SYSTÈME D'EXPLOITATION / UNIX

SUJET 6

FICHIER PARTAGÉ

Il s'agit de régler l'accès aux enregistrements d'un fichier utilisé simultanément par plusieurs programmes. Pour cela on mettra en œuvre le modèle client/serveur qui permet une plus grande sécurité que l'utilisation des verrous.

Les programmes font des requêtes de six types :

- C pour la consultation d'un enregistrement
- M pour la modification d'un enregistrement
- S pour la suppression d'un enregistrement
- A pour l'adjonction d'un enregistrement
- V pour la visualisation du fichier complet
- F pour indiquer la fin des requêtes

Le programme "**session**" permet de choisir une suite de requêtes parmi les six précédemment citées.

syntaxe : session [nom de fichier]

Un serveur est chargé d'effectuer les accès aux enregistrements afin de régler les accès concurrents lors de modification ou de suppression d'un même enregistrement. Il est lancé en arrière-plan par la commande "**servfic**" (démon) :

syntaxe : servfic -f <nom de fichier> [-d <nbre de secondes>]

La modification sera toujours constituée de 4 opérations :

- * lecture par le serveur
- * affichage du contenu courant de l'enregistrement à modifier
- * modification de ce contenu de façon interactive
- * écriture du nouveau contenu par le serveur

La suppression sera toujours constituée de :

- * lecture par le serveur
- * affichage du contenu
- * demande de validation
- * suppression effective par le serveur ou annulation de la demande de suppression

Les validations de suppression et les modifications à effectuer doivent avoir lieu dans un délai raisonnable (fixé à l'appel du serveur par l'option -d) afin d'éviter des blocages (fichier verrouillé indéfiniment par le serveur). On demande également de conserver une trace de toutes les opérations effectuées par le serveur sur le fichier.

Nous vous conseillons d'utiliser les files de messages, outils de communication inter processus d'UNIX. Prévoyez une démonstration "parlante".

P.S. : dans la mesure du possible vous ferez afficher des informations sur les ressources utilisées par vos programmes à la fin de leur exécution.