PROJET SYSTÈME D' EXPLOITATION / UNIX / LINUX

M1 Master MIAGe

Université de Lorraine

Planning projet SE 2015/16

06/11 : Présentation des sujets
11/11 au plus tard : envoyer par mail la constitution des groupes de 4 étudiants (liste des noms) + classement des 3 sujets
13/11 : attribution des sujets aux groupes
28/12 **au plus tard**: remettre les dossiers sous l'ENT
05/01: Démonstrations

LES SUJETS + recommandations pour la rédaction du dossier de projet :

SUJET 1

SIMULATION TEMPS RÉEL d'USINAGE

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.

Il s'agit de simuler cet ensemble d'opérations (usinage et remplissage) par une application temps réel. Nous vous conseillons de la découper en quatre processus lancés à partir d'un unique programme et qui s'exécuteront en parallèle.

1ère 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)

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

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

4ème 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 <nbre de pieces>]

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

Préparer une démonstration qui illustre bien la simulation selon la vitesse des différentes opérations en changeant éventuellement la syntaxe d' **entrepot.**

On s'assurera de la bonne terminaison de l'ensemble de l'application.

SUJET 2

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 se connecter sous un pseudo grâce à la commande **forum**.

Syntaxe : forum –p <pseudo>

A partir de quoi il peut faire les requêtes suivantes

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

Ces requêtes seront réalisées par le serveur si possible via des processus travaillant en parallèle pour accélérer les réponses.

Le serveur doit détruire tous les messages trop anciens.

De plus on demande de conserver une trace de toutes les opérations effectuées au sein du serveur.

SUJET 3

Gestion d'un « TWEET PRIVE »

Il s'agit de concevoir un gestionnaire de « Tweet privé », local à une machine (facilement généralisable à un réseau de machines) permettant par exemple à certains collaborateurs d'une entreprise de communiquer à l'aide de tweets. Un tweet est composé d'un texte de moins de 120 caractères et de sa date de publication. Les tweets publiés par une personne ne sont visibles que des personnes abonnées à son compte.

La gestion du « tweet » est assurée par un serveur lancé une seule fois en arrière-plan (démon) par la commande **servtw** . Ce serveur a pour fonction de gérer les comptes et les abonnements aux différents comptes des participants, de centraliser les messages publiés et de les redistribuer immédiatement aux abonnés.

Syntaxe : servtw

Avant toute utilisation du « Tweet privé» chaque utilisateur doit créer son compte sous un pseudo grâce à la commande **comptetw**

Syntaxe : comptetw –p <pseudo>

Pour pouvoir poster et recevoir des messages il doit se connecter avec la commande **tweet**.

Syntaxe : tweet -p <pseudo>

A partir de quoi il peut, d'une part, faire les requêtes suivantes :

- publier (poster) un tweet sur son compte
- s'abonner à un compte tweet (suivre une personne)
- connaître la liste des comptes tweet
- connaître les comptes tweet auxquels il est abonné
- se désabonner d'un compte tweet
- …
- se déconnecter (fermer)

D'autre part il reçoit dans une fenêtre de flux d'actualité (time line) les messages datés des comptes auxquels il est abonné.

Ces requêtes seront réalisées par le serveur si possible à l'aide de processus ou threads travaillant en parallèle pour accélérer les réponses.

Conseils pour la rédaction d'un dossier de projet

Structure d'un dossier

La réalisation d'un projet a de multiples objectifs : apprendre à résoudre des exemples plus complexes que ceux qui peuvent être abordés en travaux dirigés, approfondir une solution possible, se familiariser avec un langage de programmation, apprendre à tester un programmes mais aussi à présenter le travail réalisé. Ce dernier point est capital et mérite de s'y attarder. En effet, la rédaction d'un dossier, qu'il soit de programmation ou d'autre chose, est un travail difficile et de lui dépendra …bien évidemment la note dans le cas présent mais de manière plus générale le jugement porté sur l'ensemble du travail. En un mot, c'est le dossier qui fera vendre le programme.

Le programme résout le problème posé. Ses qualités sont sa correction (résout-il effectivement le problème posé ?), sa complétude (le fait-il dans tous les cas de figure possibles ?), son efficacité, sa facilité d'utilisation, mais aussi sa lisibilité (est-il compréhensible par d'autres ?) et sa capacité à évoluer. Certaines de ces qualités peuvent être testées (voire même automatiquement), mais c'est le dossier qui permet de comprendre

- La démarche ou la méthode suivie,
- La solution adoptée
- Les choix effectués
- La structure des objets introduits
- Ce que réalisent les fonctions définies ainsi que leur imbrication les unes par rapport aux autres
- Le texte du programme

Et aussi

- De prouver sa correction
- D'avoir une évaluation chiffrée de ses caractéristiques techniques
- De faire évoluer, de réutiliser le programme ou de corriger certaines erreurs qui ne manqueront pas de subsister et enfin
- De l'utiliser correctement

En un mot, le dossier reflète beaucoup de choses, et en particulier la compréhension du problème et le recul pris vis à vis de lui par l'auteur. La structure générale d'un dossier de projet est la suivante :

1. Introduction
2. Dossier Utilisateur
3. Dossier Concepteur
4. Dossier de Programmation
5. Dossier de Tests
6. Conclusion

Introduction

Comme tout rapport, un dossier de projet doit commencer par une introduction. Celle-ci rappelle le problème posé, sans recopier toutefois l'énoncé, en apportant plus de précision et en justifiant les interprétations faites et les grandes limitations fixées dès le départ. Dans certains cas, il faut également préciser l'environnement de développement, la liste des outils fournis, etc. La solution choisie ainsi que le plan du rapport seront ébauchés.

Dossier utilisateur

Contenu et destinataire. Comme son nom l'indique, ce dossier est destiné à un utilsateur potentiel du programme, non nécessairement informaticien et encore moins programmeur. Il doit contenir tous les renseignements nécessaires et suffisants pour une bonne utilisation ainsi que les renseignements concernant les problèmes pouvant surgir. En particulier, c'est ici qu'est décrite l'interface utilisateur.

Structure proposée

1. Appel : répertoire d'accès au programme, nom et profil de la commande à utiliser (c'est-à-dire sens de chaque paramètre, s'il y en a), outils nécessaires à l'utilisation du logiciel
2. Action : le traitement réalisé par la commande, c'est-à-dire les données indispensables, les résultats produits, les erreurs détectées
3. Déroulement de l'exécution : depuis la saisie des données jusqu'à la production des résultats. Il est nécessaire de préciser le type, la précondition (souvent il s'agit d'un intervalle de valeur), le format d'entrée et le lieu de saisie des données sur l'écran. Un exemple d'exécution aide à la compréhension. Dans le cas d'une interface sophistiquée, son utilisation doit être détaillée. En particulier, les différents menus doivent être présentés et des images des écrans de dialogue.
4. Erreurs : la liste de toutes les erreurs d'utilisation détectées par le programme, les messages correspondants ; que faire en cas d'erreur ?

Conseils

- Le dossier utilisateur doit être concis mais complet. Un exemple simple d'utilisation dans un cas normal facilite la compréhension.
- La commande proposée à l'utilisateur doit être accessible à tous et si possible auto-documentée (un appel sans paramètre rend un message d'utilisation). Si elle utilise un ou plusieurs fichiers de données, ceux-ci doivent pouvoir se trouver n'importe où dans l'arborescence des fichiers et des répertoires et pas nécessairement dans le répertoire d'appel. Si la commande crée des fichiers intermédiaires, ceux-ci doivent être supprimés en fin de commande. De façon générale, l'utilisateur doit retrouver son environnement de travail dans le même état qu'avant l'appel de la commande.

Dossier concepteur

Contenu et destinataire. Il est destiné à une personne qui veut comprendre la méthode de conception suivie, le raisonnement appliqué, les choix effectués et les raisons de ces choix. Elle doit y trouver tous les renseignements lui permettant de modifier, corriger, faire évoluer et réutiliser tout ou partie du travail.

Structure proposée. La structure peut être modulée en fonction de la complexité du problème à résoudre et de l'analyse suivie.

1. Analyse : Il est inutile de donner dans ce dossier une paraphrase de l'algorithme. Il faut détailler chaque phase du processus de développement suivi lors de la résolution du problème, en particulier le raisonnement appliqué, **les choix et les raisons des solutions adoptées**. Ainsi, le dossier concepteur final doit être fidèle à la démarche suivie. **L'utilisation de schémas est souvent une aide efficace pour expliquez vos choix.**
2. Algorithmes synthétiques.

Dossier de programmation

Contenu et destinataire. Il est destiné au programmeur qui devrait éventuellement reprendre le code. Il explique le passage de l'analyse du problème au programme.

Le traitement de l'interface utilisateur doit apparaître ici, car il dépend fortement du langage utilisé et de l'environnement de travail dont on dispose.

D'autre part, certaines modifications peuvent être apportées localement à l'analyse pour des raisons d'efficacité. Ces changements ne doivent pas être passés sous silence. Il est cependant important que le dossier concepteur reste cohérent avec le programme.

Structure proposée

1. Passage au langage de programmation : donner la représentation concrète de types d'objets introduits dans l'analyse, accompagnée d'un graphique de la représentation en mémoire.
2. Interface : Le traitement de l'interface utilisateur est souvent ignoré dans l'analyse. C'est dans ce dossier qu'il doit apparaître. A ce sujet, ne jamais oublier que le développement d'une interface utilisateur sophistiquée n'est en général pas l'objet du projet.
3. Programmes : Les textes des programmes non présentés dans ce dossier ainsi que les scripts utilisés. Les caractéristiques techniques doivent être citées : nombre le lignes de code, taille de l'exécutable, etc.

Conseils de présentation et de rédaction

- Les programmes doivent être commentés :
    1. Lexique de chaque déclaration
    2. Commentaires notant l'entrée puis la sortie des blocs décrits dans les algorithmes
    3. Eventuellement des remarques dans un corps de fonction
- Une variable locale ne doit pas être utilisée pour deux objets différents
- Les noms doivent être les mêmes dans l'analyse et dans le programme
- Les programmes font partie intégrante de ce dossier : ils doivent être présentés de la même façon (feuilles au format A4 s'ouvrant dans le même sens que les autres).

Dossier de tests

Contenu et destinataire. Bien souvent destiné au même lecteur que le dossier précédent, ce dossier doit néanmoins pouvoir être lu par n'importe qui. Il indique quels sont les tests qui ont été effectués pour garantir au maximum le bon fonctionnement du programme obtenu. Pour savoir quels sont les tests à inclure dans ce dossier, il faut identifier tous les cas possibles qui peuvent se présenter, les cas normaux et les cas d'exceptions.

Structure proposée. Deux structures sont possibles et complémentaires :

- Structure par type et par fonction
- Structure par type de test : cas normaux, limites et d'erreurs

Un test est composé d'un tableau de test et du listing correspondant. Un tableau de test a la structure suivante :

Cas précis testé	Donnée fournie	Résultat attendu	Résultat obtenu

	(valeur ou propriété)	(valeur ou propriété)	(si différent de l'attendu)

Le dossier de test doit comporter tous les cas à tester. Mais certains (faute de temps, d'espace, etc.) peuvent ne pas avoir été exécutés.

Conclusion

Tout rapport doit se terminer par une conclusion. Elle est destinée au lecteur du dossier (et tout spécialement au correcteur !). Elle doit faire un état des lieux !

Ce que fait effectivement le programme ;

Les erreurs ou incorrections détectées ;

Les cas non traités

Puis une étude prospective :

- Les évolutions possibles du programme ;
- Les parties éventuellement réutilisables

Pour terminer par quelques commentaires personnels sur la réalisation, les problèmes rencontrés, les apports du projet, etc.

Derniers conseils

Un dossier de projet est un document technique mais **il est destiné à être lu par un être humain !** Il est important que le dossier soit complet, mais aussi qu'il soit agréable à lire. D'une manière générale, il faut penser aux lecteurs, leur faciliter les différents types de lecture :

- Lecture exhaustive : ne pas se répéter mais faire des références précises (éviter les références en avant), éviter les romans fleuves et le style télégraphique, faire des phrases courtes, n'écrire que sur le recto des feuilles en laissant une marge tout autour, etc.
- Lecture en diagonale : chaque partie doit être un ensemble bien structuré, agrémenté de graphes, de schémas et d'exemples. La cohérence entre les différentes parties doit être préservée.
- Lecture directe pour aller chercher une information précise : il faut faciliter l'accès rapide (plan au début, repères en couleur, numérotation des pages, etc.), présenter un seul dossier dont toutes les pages s'ouvrent dans le même sens.

D'autre part, chaque partie est destinée à un lecteur différent, il faut en tenir compte. Enfin, il ne faut pas oublier que le dossier est le reflet de la bonne ou de la mauvaise compréhension du problème, du recul pris vis à vis du sujet et de la maîtrise de la solution de ses auteurs.