UNIVERSITE DE ROUEN

CAHIER DE RECETTES

Version :	0.1
Date :	15/01/2012
Rédigé par :	Zineb ISSAAD et Zakaria ADDI
Relu par :	Claire SMETS
Approuvé par :	

MISES A JOUR

Version	Date	Modifications réalisées
0.1	12/12/11	Création

I Introduction

I.1 Objectifs du projet

L'objectif de ce projet est d'implémenter un pilote permettant de gérer un système hybride SSD/HDD dans l'environnement Linux. Ce système doit être fiable, performant et de préférence économe en énergie. Ce pilote doit permettre aussi l'ajout et la suppression des périphériques à chaud.

La mise en place de ce pilote nécessite l'implémentation d'un protocole de gestion de cache.

Le tableau suivant spécifie les exigences fonctionnelles du produit :

Référence	Fonctionnalité globale	Acteurs	Priorité
F-Gl-10	Ouverture du périphérique	Pilote du PBV	Indispensable
F-Gl-20	Fermeture du périphérique	Pilote du PBV	Indispensable
F-Gl-30	Lecture de blocs de données	PBV et périphériques de stockage	Indispensable
F-Gl-40	Écriture de blocs de données	PBV et périphériques de stockage	Indispensable
F-Gl-50	Choix des fichiers à ne pas déplacer	Utilisateur	Optionnel
F-Gl-60	Gestion d'ajout de périphériques à chaud	Pilote	Optionnel
F-Gl-70	Gestion de suppression de périphériques à chaud	Pilote	Optionnel

1. **Ouverture du périphérique**

Notre pilote doit permettre l'ouverture du PBV. Par exemple, quand le processus client appelle la fonction "open ( )" de la bibliothèque standard *libc*, le système d'exploitation appelle notre pilote pour ouvrir le périphérique correspondant.

2. **Fermeture du périphérique**

Notre pilote doit permettre la fermeture du PBV. Par exemple, quand le processus client appelle

la fonction "release ( )" de la bibliothèque standard *libc*, le système d'exploitation appelle notre pilote pour fermer ce périphérique.

3. Lecture de blocs de données

Quand le processus client veut lire un fichier en utilisant la fonction "read ( )" de la bibliothèque standard *libc,* le système d'exploitation appelle notre pilote pour retourner les blocs autorisés en lecture correspondant au fichier demandé.

Dans le cas où le fichier demandé n'existe pas, le système ne renvoie rien.

4.Écriture de blocs de données

Quand le processus client veut écrire dans un fichier en utilisant la fonction "write ( )" de la bibliothèque standard *libc,* le système d'exploitation appelle notre pilote pour envoyer le fichier correspondant. Dans le cas où ce dernier n'existe pas, il sera créé.

5. Choix des fichiers à ne pas déplacer

Dans le cas où le SSD (HDD) est plein le système demande à l'utilisateur de spécifier certains fichiers qui ne seront pas déplacés vers le HDD (SSD). Ce choix est indépendant du nombre ou de la fréquence d'accès.

6. Gestion d'ajout de périphériques à chaud

L'utilisateur peut ajouter au système un périphérique à chaud. Ce dernier, est ajouté au PBV si l'utilisateur est d'accord, dans ce cas, toutes les données contenant le périphérique seront écrasées. Sinon, ce périphérique n'est pas pris en compte.

Le périphérique ajouté est configuré pour être utilisé avec le pilote.

7. Gestion de suppression de périphériques à chaud

Quand le processus client veut retirer le périphérique à chaud et que les opérations de maintenances ne sont pas achevées, le système doit gérer cette situation.

I.2 Liste des objets à tester

L'objet que nous allons tester est le module noyau développé que nous avons appelé « Pilote ».

II Documents applicables et de référence

Ce cahier de recette est établi en utilisant :

- L'appel d'offre : Projet1-2-3.pdf
- Le document de la Spécification Technique des Besoins (STB)
- Le document Architecture Du Logicielle (DAL)

III Terminologie et sigles utilisés

- **Module noyau :** Programme qui peut être chargé dynamiquement dans le noyau.

-**SSD :** Solid State Disk, périphérique de stockage qui utilise de la mémoire flash. Les performances sont élevées et les consommations d'énergie basses mais le coût par Go est élevé.

-**HDD :** Hard Drive Disk, périphérique de stockage de masse. Il consomme plus et ses performances sont moindres qu'un SSD mais le prix d'acquisition est beaucoup moins élevé.

-**PBV :** Périphérique Bloc Virtuel. Il s'agira ici d'un fichier représentant deux disques (SSD et HDD). Il est cependant possible qu'il ne soit composé que du HDD.

-**SGF :** Un Système de Gestion de Fichiers "est une façon de stocker les informations et de les organiser dans des fichiers". (Wikipedia)

-**Pilote :** Programme gérant les opérations entre un périphérique et le reste du système.
-**Mémoire cache :** Mémoire volatile d'accès rapide. Les données ne sont pas liées à un processus en cours d'exécution.

-**Swap :** Zone mémoire réservée sur l'espace de stockage de masse et/ou sur le PBV servant de mémoire d'appoint à la mémoire principale.

-**Fichier :** Sous les systèmes d'exploitation avec un système de noyau linux, tout est "fichier" : processus, partitions, etc.

-**Libc :** bibliothèque standard de linux permettant les appels système par le processus client.

Environnement de test

-Les tests seront réalisés dans une salle de TP (M1 SSI)

-Le matériel nécessaire pour réaliser les tests est :

- ✓ Un HDD
- ✓ Un SSD de taille 80 Go.

-Les produits devant être installés sur la machine sont :

- ✓ Le système Linux (Ubuntu version 11.04)
- ✓ Des contrôleurs SATA et/ou IDE

-Les tests seront réalisés sur des fichiers contenant des données quelconques.

Responsabilités

-La réalisation des tests seront effectués par :

- ✓ Zineb ISSAAD
- ✓ Zakaria ADDI

-Pour pouvoir effectuer ces tests, nous devons avoir à notre disposition :

- ✓ Le pilote développé ;
- ✓ Les fichiers sur lesquels s'effectuent les tests ;
- ✓ Les procédures de tests.

-Les anomalies seront corrigées par les développeurs.

Stratégie de tests

La stratégie de tests sera comme suit :

1) Le pilote développé sera remis aux testeurs.
2) Pour chaque procédure de test, tester tous les jeux de test dans d'ordre croissant de leur numéro.
3) Rédiger un rapport contenant les résultats retournés par les tests effectués
4) Le rapport sera transmis aux développeurs
5) Les développeurs corrigeront les anomalies détectées

6) Le pilote sera retransmis aux testeurs pour refaire un autre test
7) Recommencer à partir du point 2), [Tenir compte du temps attribué à ces tests]
8) Les tests seront terminés quand toutes les anomalies seront corrigées

Gestion des anomalies

Le rapport rédigé lors de la réalisation des tests contient ce qui suit :

- ✓ Le numéro de la procédure
- ✓ Le numéro de test
- ✓ Les anomalies détectées pour chaque test
- ✓ A chaque fois qu'un test est refait, mentionner l'état de l'anomalie : ok /no ok.

Procédures de test

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Ouverture du périphérique en lecture				
Procédure n° <F-FN-10>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Utiliser la fonction « open ( ) » pour ouvrir un périphérique	Ouverture du périphérique	Fonctionnement	
2	Utiliser la fonction « open ( ) » pour ouvrir un périphérique qui n'existe pas	Rien ne se passe	Fonctionnement	
4	Sauvegarder le fichier « test » sur le HDD et sur le PBV puis, effectuer une opération sur le PBV ensuite sur le HDD pour ouvrir ce fichier.	Le temps d'accès au PBV est inférieur au temps d'accès au HDD	Performance : rapidité	
5	Sauvegarder le fichier « test » sur le HDD et sur le SSD puis, effectuer un « open ( ) » sur le HDD ensuite sur le SSD pour ouvrir ce fichier.	Le contenu du fichier « test» ouvert à partir du SSD et HDD est le même.	Fiabilité	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

- ✓ Créer un fichier « test» qui contient une suite de « 0 »

J1 :

- ✓ Attribuer à ce fichier que les droits de lecture.

J4 :

- ✓ Mettre le fichier « test » sur le HDD et sur le PBV.

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Ouverture du périphérique en écriture				
Procédure n° <F-FN-20>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Utiliser la fonction « open ( ) » pour ouvrir un périphérique	Ecrire sur le périphérique	Fonctionnement	
2	Utiliser la fonction « open ( ) » pour ouvrir un périphérique qui n'existe pas	Rien ne se passe	Fonctionnement	

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Fermeture du périphérique				
Procédure n° <F-FN-30>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Utiliser la fonction « release ( ) » pour fermer le fichier « test »	-Décrémente le compteur d'utilisateur du PBV - Si le compteur est à 0 fermer le PBV	Fonctionnement	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

J1 :

✓ Le fichier « test » est ouvert

.

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Lecture de blocs de données				
Procédure n° <F-FN-40>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Utiliser la fonction « read ( )» sur le fichier « test» autorisé en lecture	Le système retourne les blocs correspondants	Fonctionnement	
2	Utiliser la fonction « read ( )» sur un fichier qui n'existe pas	Le système ne retourne rien	Fonctionnement	
3	Utiliser la fonction « read ( )» sur le fichier « test » non autorisé en lecture	Le système ne retourne rien	Sécurité	

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Ecriture de blocs de données				
Procédure n° <F-FN-50>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Utiliser la fonction « write ( )» pour écrire sur le fichier « test » autorisé en écriture	Ouverture du fichier correspondant	Fonctionnement	
2	Utiliser la fonction « write ( )» pour écrire sur un fichier qui n'existe pas	Création puis ouverture du fichier.	Fonctionnement	
3	Utiliser la fonction « write ( )» pour écrire sur le fichier « test » non autorisé en écriture	Le système ne retourne rien	Sécurité	

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Choix des blocs à ne pas déplacer				
Procédure n° <F-FN-60>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Sauvegarder les blocs du fichier « test » sur le SSD, mais ce dernier est plein.	Le système demande de choisir les fichiers à ne pas déplacer vers le HDD	Fonctionnement	
2	Choix des fichiers à ne pas déplacer du SSD : fichier « test1 » et « test2 »	-Le fichier « test3 » est déplacé vers le HDD - Les fichiers « test », « test1 » et « test2 » se trouvent sur le SSD	sécurité	
3	Sauvegarder les blocs du fichier « test » sur le HDD, mais ce dernier est plein.	Le système demande de choisir les fichiers à ne pas déplacer vers le SSD »	Fonctionnement	
4	Choix des fichiers à ne pas déplacer du HDD : fichier « test1 » et « test2 »	-Le fichier « test3 » est déplacé vers le SSD - Les fichiers « test », « test1 » et « test2 » se trouvent sur le HDD	sécurité	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

✓ En plus du fichier « test », nous allons créer trois autres fichiers qui sont : « test1», « test2» et « test3»

J1 :

✓ Les fichiers « test1», « test2» et « test3» se trouve sur le SSD

✓ Le SSD est plein

J2 :

✓ Le fichier « test1» et « test2» sont dans le SSD

J3 :

✓ Les fichiers « test1», « test2» et « test3» se trouve sur le HDD

✓ Le HDD est plein

J4 :

✓ Le fichier « test1» et « test2» sont dans le HDD

Objet testé : pilote			**Version :** <0.1>	
Objectif de test : Déplacement de blocs du HDD vers le SSD				
Procédure n° <F-FN-80>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Faire appel au fichier « test» se trouvant sur le HDD plusieurs fois soit pour lire, écrire …	Le fichier « test» sera déplacé vers sur le SSD	Fonctionnement	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

J1 :

✓ Le fichier « test» se trouve sur le HDD

Objet testé : pilote	**Version :** <0.1>
Objectif de test : Gestion d'ajout de périphériques à chaud	
Procédure n° <F-FN-90>	

N°	Actions	Résultats attendus	Exig.	OK/ NOK
1	Brancher un périphérique	La détection du périphérique ajouté	Fonctionnement	
2	Mettre dans le périphérique ajouté des données non gérées par le PBV	Le système demande si l'utilisateur veut toujours continuer.	sécurité	
3	Nous continuons	-Les données du périphérique ajouté sont écrasées -Le volume du PBV est augmenté du volume du périphérique ajouté	Fonctionnement	
4	Nous refusons de continuer	Le périphérique ajouté n'est pas pris en compte.	Fonctionnement	
5	Enlever le pilote permettant de configurer le périphérique ajouté	Le périphérique ajouté n'est pas détecté	Fonctionnement	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

J1 :

✓ Le périphérique à ajouter

✓ La configuration de ce périphérique

J2 :

✓ Le périphérique ajouté contient des données

Objet testé : pilote		**Version :** <0.1>		
Objectif de test : Gestion de suppression de périphériques à chaud				
Procédure n° <F-FN-100>				
N°	**Actions**	**Résultats attendus**	**Exig.**	**OK/ NOK**
1	Démonter le périphérique déjà détecté par le système, sachant que toutes les opérations de maintenances sont terminées	-La synchronisation des données -Le périphérique est démonté du système - Le volume du PBV est diminué du volume du périphérique démonté	Fonctionnement	
2	Démonter le périphérique pendant l'opération d'écriture (par exemple) sur le fichier « test»	-Le périphérique est démonté du système - Le volume du PBV est diminué du volume du périphérique démonté	Fonctionnement	

Jeux de données de test

Afin de pouvoir réaliser ces jeux de test, nous allons mettre à notre disposition ce qui suit :

J1 :

✓ Le périphérique à démonter est détecté par le système

✓ Toutes les opérations de maintenances sont terminées

J2 :

✓ Le fichier « test» est ouvert en écriture