Déroulement SCRUM du Projet Ordonnanceur Multitâche sous Linux

1- Structure de l’Équipe SCRUM

Dans le cadre de SCRUM, l'équipe est auto-organisée favorisant la polyvalence et la responsabilité collective. Ce projet applique Scrum dans une version adaptée à un contexte académique et à une équipe réduite. Les rôles ne sont pas exclusifs , chaque membre peut contribuer au développement pour maximiser l'efficacité.

Membre	**Rôle**	Mission
Wiem hamzaoui	Scrum master(SM)	-Organiser et faciliter les cérémonies Scrum. -Éliminer les obstacles et aider l’équipe à rester productive. -Veiller au respect des règles Scrum et encourager l’amélioration continue.
Nour chaker	Product owner(PO)	-Définir la vision du produit et rédiger les User Stories. -Prioriser le Product Backlog selon la valeur métier. -Valider les fonctionnalités livrées et s’assurer qu’elles répondent aux besoins.
Mazen khoualdi Nour Chaker Wiem hamzaoui	Development Team(DEV)	-Concevoir et implémenter les fonctionnalités du produit selon les User Stories. -corriger les bugs. - faire des tests de fonctionnement. -Documenter, intégrer et maintenir le code pour garantir un produit évolutif et bien structuré.

2. Product Backlog (PB)

ID	Élément du Backlog	Description	Priorité	Critere d'acceptation
1	Lecture et parsing du fichier de configuration	Le système doit charger un fichier décrivant les processus, en acceptant commentaires et lignes vides.	Haute	Le parser ignore les lignes vides et les commentaires (# et //). Chaque ligne valide crée une structure `Process`. Les erreurs de format sont gérées proprement.
2	Politique FIFO	Permettre l'exécution des processus selon une politique d'ordonnancement FIFO.	Haute	FIFO est compilée en `.so` et chargée dynamiquement. Les processus s'exécutent selon l'ordre d'arrivée. Les résultats sont affichés correctement en console.
3	Politique Round Robin	Permettre l'ordonnancement des processus selon la politique Round Robin avec un quantum paramétrable.	Haute	La valeur du quantum est saisie par l'utilisateur. La politique est chargée dynamiquement (.so). La simulation respecte le quantum et le préemptif.
4	Politique de priorité préemptive	Implémenter l'ordonnancement par priorité préemptive.	Haute	Interruption si un processus de priorité supérieure arrive. Chargement dynamique. Résultats corrects sur la timeline.
5	Chargement dynamique des politiques d'ordonnancement	Le système doit charger automatiquement les politiques disponibles dans le dossier `policies/`.	Très haute	Le menu des politiques s'adapte automatiquement aux `.so` présents. Aucune recompilation n'est nécessaire pour ajouter une politique.

6	menu interactif	Permettre à l'utilisateur de sélectionner une politique, configurer les paramètres, et lancer la simulation.	Moyenne	Menu clair : choix de politique, quantum, fichier.
7	Politique MLQ (Multi-Level Queue) priorité statique	Implémenter MLQ avec priorités statiques.	Haute	Les processus sont affectés à une queue selon priorité statique. Les queues sont traitées dans l'ordre (niveau 1 → n).
8	MLQ avec priorité dynamique (aging)	Permettre l'ordonnancement multi-niveaux avec mécanisme d'aging afin de prévenir la famine des processus.	Haute	La priorité d'un processus diminue après un temps d'attente défini. Aucun processus ne reste indéfiniment bloqué. Simulation correcte
9	Automatisation de la compilation et du build du projet	Le Makefile doit compiler le programme principal, générer les bibliothèques partagées des politiques, gérer les dépendances système et fournir des cibles standards facilitant l'installation et le nettoyage du projet.	Très Haute	- La commande `make` compile l'ensemble du projet sans erreur - Le binaire principal est généré correctement - Chaque politique est compilée en bibliothèque partagée (`.so`) - La commande `make clean` supprime tous les fichiers générés - La commande `make install` permet l'installation du programme sans nécessiter de modification du code
10	Affichage console des résultats	Fournir un affichage clair des résultats : temps d'attente, turnaround, timeline.	Moyenne	Affichage du Gantt textuel. Calculs corrects des métriques.
11	Affichage graphique des résultats	Générer un diagramme de Gantt graphique avec GTK.	Basse à moyenne	Visualisation correcte du déroulement de la simulation.

				Interface stable et exploitable.
12	IHM graphique pour suivre la simulation	Interface GTK permettant de lancer une simulation et voir l'évolution en temps réel.	Basse	Bouton de lancement. Visualisation dynamique. Mise à jour fluide de l'affichage.
13	Documentation	Installation, technique, Scrum	Haute	Conforme au cahier des charges
14	Licence	Choix du licence + Fichier de licence	Moyenne	Fichier `LICENSE` à la racine

3. Planification des sprints

Le tableau suivant présente la planification des sprints tout au long de notre période de réalisation

N°	Sprint	Estimation
1	Fondations du Système	1 semaine
2	Implémentation des Politiques Minimales	1 semaine
3	Finalisation et Fonctionnalités Avancées	1 semaines

NB: Les sprints décrits correspondent aux phases actives de développement du projet. Les périodes intermédiaires ont été consacrées à l'analyse des exigences, à la recherche, à la documentation et aux ajustements techniques.

Sprint 1 : Fondations du Système (Semaine 1)

Objectif : Établir les bases architecturales pour une implémentation stable, en se concentrant sur le chargement des données et les outils de build.

Tâches Détaillées :

Tâche	Description	Responsable	Effort Estimé
T1.1	Analyse approfondie du cahier des charges.	Toute l'équipe	3

T1.2	Définition de l'arborescence et l' architecture du projet.	Nour	2
T1.3	Création du fichier des processus.	Wiem	3
T1.4	Création des structures de données pour les processus (struct Process avec PID, arrival_time, burst_time, priority)	Mazen	2
T1.5	Lecture et Parsing du Fichier des processus	Wiem	3
T1.6	Configuration du Makefile	Nour	2
T1.7	main.c pour Validation Initiale	Mazen	

DOD (Def of Done):

- Module de lecture et parsing du fichier de configuration.
- Structures de données prêtes .
- Makefile fonctionnel et automatisé.
- Documentation initiale : Spécification du format de configuration et guide de compilation.

Sprint 2 : Implémentation des Politiques Minimales (Semaine 2)

Objectif : Intégrer les algorithmes d'ordonnancement requis, en assurant une modularité pour les extensions futures.

Tâches Détaillées :

Tâche	Description	Responsable	Effort Estimé

T1.1	Implémentation de l'algorithme FIFO avec simulation temporelle.	Nour	5
T1.2	Implémentation de Round-Robin avec gestion du quantum et context switching.	Wiem	5
T1.3	Implémentation de la priorité préemptive avec préemption basée sur événements.	Nour	5
T1.4	Conception d'une fonction handler commune pour appeler les politiques.	Wiem	3
T1.5	Lecture dynamique des politiques depuis le répertoire /policies (scan de fichiers).	Mazen	3
T1.6	Développement d'un menu interactif en console pour sélectionner les politiques.	Mazen	3
T31.7	Écriture et exécution de tests unitaires .	Tous	5
T1.8	Mise à jour de la documentation technique sur les choix algorithmiques.	Nour	2

DOD (Def of Done):

- Implémentations fonctionnelles des politiques FIFO, RR et Priorité Préemptive.
- Répertoire /policies avec modules séparés pour chaque algorithme.
- Menu dynamique et interface console basique.
- Tests unitaires couvrant au moins 80% des cas.

Sprint 3 : Finalisation et Fonctionnalités Avancées (Semaine 3)

Objectif : Stabiliser le produit, ajouter des fonctionnalités avancées si la vélocité le permet, et préparer la livraison finale.

Tâches Détaillées :

Tâche	Description	Responsable	Effort Estimé
T1.1	Implémentation du Multi-Level Queue Scheduling avec allocation de queues.	Mazen	8
T1.2	Ajout de l'aging pour ajuster dynamiquement les priorités.	Mazen	5
T1.3	Développement d'une interface graphique (e.g., visualisation des timelines).	Wiem	8
T1.4	Support du chargement dynamique via libdl, utilisant les fonctions dlopen, dlsym et dlclose.	Wiem	5
T1.5	Rédaction du rapport final et du guide d'utilisation en anglais.	Nour	5
T1.6	Packaging final (archive ZIP/tar avec tous les fichiers).	Nour	2

DOD (Def of Done):

- Fonctionnalités avancées intégrées .
- Documentation complète : Guide utilisateur en anglais, PDF sur les choix techniques.
- Projet stable, compilable et testable.
- Archive finale prête pour soumission.

4. Daily Scrum

Des points d’avancement courts et réguliers ont été organisés afin d’inspecter l’avancement du travail, identifier les obstacles et ajuster les tâches à venir. Leur fréquence a été adaptée aux contraintes académiques de l’équipe.

5. Sprint Review

À la fin de chaque sprint, une réunion de revue (30min-1h) pour démontrer les livrables au PO . L'équipe présente le travail accompli, mesure l'avancement par rapport au Sprint Goal, et collecte des feedbacks pour ajuster le Backlog.

Processus :

- Démonstration interactive du logiciel.
- Vérification des User Stories complétées.
- Validation par le PO : Acceptation ou retour pour le prochain sprint.