Rapport de projet LP25

Liste des membres :

- ISLAM Abir – TC3
- MELLOUK Mohamed-Amine – TC3
- FALLANI Issam – TC3

Automne 2025

Contexte du projet

Le projet LP25 consistait à concevoir un programme de gestion de processus pour systèmes Linux, capable de fonctionner en local et, dans une certaine mesure, sur des hôtes distants. L'objectif principal était de fournir une interface interactive inspirée de l'outil htop, permettant d'afficher dynamiquement la liste des processus actifs, de visualiser des informations détaillées (PID, utilisateur, consommation CPU et mémoire, temps d'exécution, priorité, état du processus, etc.) et d'interagir avec ces processus via des actions telles que l'arrêt, la mise en pause ou le redémarrage. Ce type d'outil s'inscrit dans une démarche d'administration système moderne, où la supervision et le contrôle granulaire des processus constituent des compétences essentielles pour tout ingénieur système.

Le projet repose sur des mécanismes standards de l'administration système Linux, notamment l'utilisation du système de fichiers virtuel /proc pour la récupération en temps réel des informations sur les processus, l'envoi de signaux Unix (SIGTERM, SIGKILL, SIGSTOP, SIGCONT) pour l'interaction avec ceux-ci, ainsi que l'utilisation de bibliothèques comme ncurses pour la gestion de l'interface utilisateur en mode texte. L'architecture devait également prévoir une extensibilité vers la gestion distante via le protocole SSH, permettant ainsi une administration multi-machines depuis une console unique. Cette approche multi-couches impose une réflexion approfondie sur la séparation des responsabilités et l'abstraction des couches système.

État des lieux des compétences du groupe

Avant le début du projet LP25, le groupe possédait un niveau globalement intermédiaire en langage C. Les notions fondamentales (structures, pointeurs, allocation dynamique, compilation séparée, gestion de la mémoire) étaient acquises à travers les Cours, TD et TP précédents. En revanche, la programmation système sous Linux constituait un domaine encore peu maîtrisé. La compréhension théorique des mécanismes du noyau et des appels système restait limitée, ce qui constituait un frein pour l'implémentation de fonctionnalités avancées.

Les membres du groupe avaient peu ou pas d'expérience préalable avec :

- la lecture et l'interprétation du contenu de /proc (parsing des fichiers stat, status, cmdline),
- la gestion avancée des processus (signaux, états, priorités, relations parent-enfant),
- la conception d'interfaces interactives avec ncurses et la gestion des événements clavier,
- la communication distante via SSH et l'utilisation de bibliothèques telles que libssh,
- la gestion de l'architecture multi-thread pour un rafraîchissement asynchrone.

Ces lacunes ont permis d'anticiper dès le début du projet des difficultés, notamment sur l'architecture globale du programme, la robustesse du parsing des données système, et sur la compréhension fine des mécanismes internes du système Linux. Il était donc nécessaire de prévoir une phase d'apprentissage conséquente avant la phase d'implémentation proprement dite.

Répartition des tâches : planification

Lors de la phase de planification, les tâches suivantes ont été identifiées :

- Analyse du sujet et des objectifs du projet (4 heures)
- Recherche documentaire sur /proc et la gestion des processus Linux (6 heures)
- Conception de l'architecture du programme (modules, fichiers, responsabilités) (6 heures)
- Implémentation de la récupération des processus locaux (8 heures)
- Implémentation de l'interface utilisateur avec ncurses (10 heures)
- Implémentation des actions sur les processus (signaux) (6 heures)
- Tentative de mise en place de la communication distante via SSH (6 heures)
- Tests, débogage et corrections (8 heures)

Les tâches ont été réparties de manière relativement équilibrée entre les membres, avec une spécialisation progressive selon les affinités techniques.

Répartition des tâches : réalisation

Dans la pratique, la répartition réelle des tâches a légèrement différé de la planification initiale, notamment en raison de l'émergence de difficultés techniques imprévues et de la courbe d'apprentissage plus importante que prévu :

- **ISLAM Abir :** travail principal sur l'interface ncurses, gestion de l'affichage dynamique, rafraîchissement périodique et interaction utilisateur avec système de navigation par onglets (temps réel estimé : ~14 heures)
- **MELLOUK Mohamed-Amine :** récupération et traitement des informations des processus via /proc, structuration des données en structures C optimisées, calcul des métriques de performance (temps réel estimé : ~12 heures)
- **FALLANI Issam :** gestion des signaux et actions sur les processus, vérification des permissions, participation à l'architecture générale et au débogage intensif (temps réel estimé : ~12 heures)

La partie concernant la gestion distante via SSH a été abordée collectivement, mais n'a pu être finalisée faute de temps et de maîtrise suffisante des bibliothèques nécessaires (libssh). Des tests préliminaires ont néanmoins été effectués, permettant d'établir une base de travail pour d'éventuelles améliorations futures du projet.

Difficultés rencontrées

Compréhension des objectifs

Une première difficulté a concerné la portée exacte des fonctionnalités attendues, en particulier concernant la gestion distante. Les consignes restaient volontairement ouvertes, ce qui a rendu difficile l'évaluation du niveau de complétude attendu.

Pour pallier ce manque, le groupe s'est appuyé sur l'analyse d'outils existants (htop, top) et sur des échanges entre membres afin de définir un périmètre réaliste. Avec le recul, une clarification plus précoce avec l'enseignant aurait permis de mieux cibler les priorités du projet.

Réalisation des objectifs

Plusieurs difficultés techniques ont été rencontrées lors de l'implémentation, nécessitant des ajustements importants dans l'approche initiale :

- **Lecture de /proc :** les premiers essais produisaient des données incohérentes ou incomplètes, notamment concernant le calcul du pourcentage CPU qui nécessite une compréhension fine du temps système et utilisateur. Le problème a été résolu en restructurant le code de parsing, en implémentant une gestion robuste des erreurs, et en vérifiant systématiquement la validité et l'existence des fichiers lus.
- **Interface ncurses :** des problèmes de rafraîchissement et de clignotement de l'affichage ont été observés, notamment lors du redimensionnement de la fenêtre terminal. Une meilleure séparation entre la logique métier et l'affichage, couplée à l'utilisation de fenêtres ncurses (windows) séparées, a permis d'améliorer significativement la stabilité et la fluidité de l'interface.
- **Gestion des signaux :** certaines actions sur les processus ne produisaient pas l'effet attendu, voire provoquaient des comportements inattendus. L'origine du problème était liée à des droits insuffisants (nécessité d'exécution en root), à une mauvaise interprétation de l'état du processus (zombie, stopped), ou à des processus appartenant à d'autres utilisateurs.
- **Performances :** avec un grand nombre de processus actifs (>200), le rafraîchissement devenait perceptiblement lent. L'optimisation des algorithmes de tri et de filtrage a permis d'améliorer les temps de réponse.

Ces problèmes auraient pu être anticipés par des tests unitaires plus précoces, une exploration plus approfondie de la documentation système (man pages, documentation du noyau), et une phase de prototypage sur des cas limites (processus zombies, contraintes de permissions, charge système élevée).

Proposition d'une amélioration

Le modèle de code actuel repose sur une architecture relativement monolithique. Une piste d'amélioration consisterait à adopter une architecture plus modulaire, en séparant clairement :

- la couche d'acquisition des données système,
- la logique métier,
- la couche d'interface utilisateur.

L'introduction d'abstractions plus claires et éventuellement l'utilisation de structures de données plus adaptées améliorerait la maintenabilité, la lisibilité du code et faciliterait l'ajout de nouvelles fonctionnalités, notamment la gestion distante.

Analyse des résultats

Le programme développé permet d'afficher correctement les processus locaux avec un rafraîchissement en temps réel, de consulter plusieurs informations pertinentes (PID, utilisateur, état, consommation ressources, ligne de commande) et d'effectuer certaines actions de base sur les processus (pause, arrêt, kill, reprise). L'interface utilisateur, bien que minimaliste, offre une expérience d'utilisation fluide et intuitive comparable aux outils standards. Bien que toutes les fonctionnalités envisagées n'aient pas été pleinement réalisées (notamment la partie réseau complète), le résultat reste cohérent avec les objectifs pédagogiques du projet et constitue une base fonctionnelle solide.

Le projet a permis au groupe de consolider ses compétences en programmation C, particulièrement concernant la gestion de la mémoire, le parsing de fichiers texte complexes, et la manipulation de structures de données. Plus important encore, il a permis de découvrir concrètement la programmation système sous Linux, l'utilisation des API système (signaux, processus), et les bonnes pratiques d'architecture logicielle pour des applications système critiques.

Retour d'expérience

Le déroulement réel du projet a mis en évidence un écart notable entre la planification initiale et la réalisation effective. La principale source de blocage a été la sous-estimation du temps nécessaire à la compréhension des mécanismes système.

Avec le recul, il aurait été plus efficace de :

- consacrer davantage de temps à la phase de conception,
- prioriser plus strictement les fonctionnalités essentielles,
- mettre en place des points de validation intermédiaires.

Ce retour d'expérience souligne l'importance de la planification réaliste, de la communication au sein du groupe et de l'anticipation des difficultés techniques. Ces enseignements seront directement réutilisables dans de futurs projets d'ingénierie.

Conclusion

Le projet LP25 a constitué une expérience formatrice, tant sur le plan technique qu'organisationnel. Malgré certaines limites fonctionnelles, le travail réalisé répond aux objectifs principaux et a permis au groupe de développer des compétences essentielles en programmation système et en travail collaboratif.

Ce projet a également mis en lumière l'importance de l'anticipation, de la rigueur et du retour critique sur son propre travail, éléments indispensables dans la formation et le futur métier d'ingénieur.