RAPPORT TECHNIQUE

Projet LP25 – Moniteur de processus Linux

ISLAM Abir • MELLOUK Mohamed-Amine • FALLANI Issam

Université de Technologie de Belfort-Montbéliard (UTBM)

Janvier 2026

Table des matières

RAPPORT TECHNIQUE

Projet LP25 – Moniteur de processus Linux

Auteurs :
- ISLAM Abir
- MELLOUK Mohamed-Amine
- FALLANI Issam

Formation : Licence Professionnelle 25
Établissement : Université de Technologie de Belfort-Montbéliard (UTBM)
Date : Janvier 2026

Table des matières

1. Introduction

1.1 Contexte

Dans le cadre de notre formation en Licence Professionnelle 25, nous avons été amenés à réaliser un projet de développement système sous Linux. L'objectif était de créer un outil de surveillance et de gestion de processus, inspiré de l'utilitaire **htop**, tout en y ajoutant des fonctionnalités de supervision réseau.

1.2 Objectifs pédagogiques

Ce projet vise à consolider nos compétences en : - Programmation système en langage C - Manipulation du système de fichiers `/proc` sous Linux - Développement d'interfaces textuelles avec ncurses - Communication réseau via SSH - Gestion de projet informatique et travail collaboratif

1.3 Périmètre du projet

Le projet se décompose en deux parties principales : - **Mode local** : surveillance des processus de la machine hôte - **Mode réseau** : surveillance simultanée de plusieurs machines distantes

---

2. Cahier des charges

2.1 Exigences fonctionnelles

2.1.1 Mode local

L'application doit permettre de : - Afficher la liste des processus en cours d'exécution - Présenter les informations suivantes pour chaque processus : - PID (Process ID) - Nom du processus - Utilisateur propriétaire - État du processus (Running, Sleeping, Zombie, etc.) - Utilisation CPU (en %) - Utilisation mémoire (en %) - Temps CPU cumulé - Mettre à jour l'affichage en temps réel - Permettre la navigation dans la liste des processus - Implémenter une fonction de recherche - Gérer les processus via des signaux : - SIGSTOP (pause) - SIGCONT (reprise) - SIGTERM (arrêt propre) - SIGKILL (terminaison forcée)

2.1.2 Mode réseau

L'application doit permettre de : - Se connecter à plusieurs machines distantes via SSH - Afficher les processus de chaque machine dans des onglets distincts - Naviguer entre les onglets (machines) - Gérer les processus distants de la même manière que les processus locaux - Gérer les erreurs de connexion et les déconnexions

2.2 Exigences non fonctionnelles

2.2.1 Performance

- Rafraîchissement fluide de l'interface (latence < 1 seconde)
- Gestion efficace de la mémoire (pas de fuite mémoire)
- Support de milliers de processus sans dégradation des performances

2.2.2 Sécurité

- Authentification sécurisée pour les connexions SSH
- Protection des identifiants (pas de stockage en clair dans le code)
- Validation des permissions avant envoi de signaux aux processus
- Fichier de configuration avec permissions restrictives (600)

2.2.3 Utilisabilité

- Interface claire et intuitive
- Raccourcis clavier cohérents
- Messages d'erreur explicites
- Documentation complète

2.2.4 Portabilité

- Compatible avec les distributions Linux majeures (Debian, Ubuntu, CentOS, etc.)
- Pas de dépendances propriétaires

2.3 Contraintes techniques

- **Langage** : C (norme C99 ou supérieure)
- **Interface** : ncurses pour l'affichage en terminal
- **Réseau** : libssh pour les connexions SSH
- **Compilation** : Makefile compatible avec GNU Make

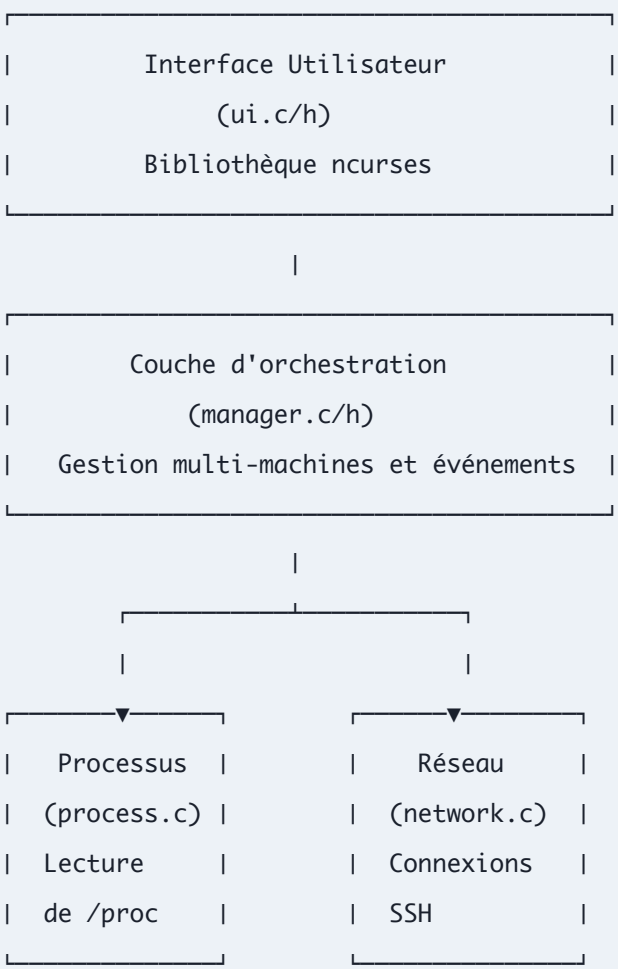
- **Dépendances** : build-essential, libncurses5-dev, libssh-dev

---

3. Analyse et conception

3.1 Architecture générale

Le projet suit une architecture modulaire en couches :



3.2 Modules principaux

3.2.1 Module main.c

- Point d'entrée du programme
- Parsing des arguments de ligne de commande

- Initialisation des modules
- Boucle principale d'événements

3.2.2 Module process.c/h

- Lecture du système de fichiers `/proc`
- Parsing des fichiers `/proc/[pid]/stat`, `/proc/[pid]/status`, etc.
- Calcul des métriques (CPU, mémoire)
- Envoi de signaux aux processus
- Structures de données pour représenter un processus

Structure principale :

```
typedef struct {
    pid_t pid;
    char name[256];
    char user[64];
    char state;
    float cpu_percent;
    float mem_percent;
    unsigned long long cpu_time;
    // ... autres champs
} Process;
```

3.2.3 Module network.c/h

- Gestion des connexions SSH
- Authentification (mot de passe, potentiellement clé SSH)
- Exécution de commandes distantes
- Parsing des résultats
- Gestion des erreurs de connexion

Structure principale :

```
typedef struct {
    char name[64];
    char ip[64];
    int port;
    char username[64];
    char password[256];
    ssh_session session;
    ssh_channel channel;
    bool connected;
} RemoteHost;
```

3.2.4 Module manager.c/h

- Orchestration des sources de processus (local + distants)
- Agrégation des données
- Gestion de la liste des hôtes
- Coordination des rafraîchissements

3.2.5 Module ui.c/h

- Initialisation de ncurses
- Affichage de l'interface (en-tête, liste, barre d'état)
- Gestion des onglets pour le mode multi-machines
- Gestion des événements clavier
- Mise à jour dynamique de l'affichage
- Fenêtres de dialogue (confirmation, recherche, aide)

3.3 Flux de données

3.3.1 Mode local

```
1. Lecture de /proc
2. Parsing des fichiers
3. Calcul des métriques
4. Stockage dans structures Process
5. Affichage via ncurses
6. Attente événement clavier ou timer
7. Retour à l'étape 1
```

3.3.2 Mode réseau

```
1. Connexion SSH aux hôtes distants
2. En parallèle pour chaque hôte :
   a. Exécution de commandes de collecte
   b. Récupération des résultats
   c. Parsing des données
3. Agrégation locale + distantes
4. Affichage par onglets
5. Gestion navigation entre onglets
6. Retour à l'étape 2
```

3.4 Gestion de /proc

Le système de fichiers `/proc` fournit les informations sur les processus. Voici les principaux fichiers utilisés :

Fichier	Informations
`/proc/[pid]/stat`	État, temps CPU, mémoire virtuelle
`/proc/[pid]/status`	État détaillé, UID, mémoire
`/proc/[pid]/cmdline`	Ligne de commande complète
`/proc/stat`	Statistiques système (pour calcul CPU)
`/proc/meminfo`	Informations mémoire globale

Algorithme de calcul du CPU :

```
1. Lire /proc/[pid]/stat à t0
2. Lire /proc/stat à t0 (pour total CPU système)
3. Attendre Δt
4. Lire /proc/[pid]/stat à t1
5. Lire /proc/stat à t1
6. cpu_percent = (Δcpu_process / Δcpu_total) * 100
```

3.5 Gestion des connexions SSH

Pour le mode réseau, nous utilisons la bibliothèque **libssh** :

Flux de connexion :

1. ssh_new() - Créer une session
2. ssh_options_set() - Configurer host, port, user
3. ssh_connect() - Établir la connexion
4. ssh_userauth_password() - Authentification
5. ssh_channel_new() - Créer un canal
6. ssh_channel_open_session() - Ouvrir la session
7. ssh_channel_request_exec() - Exécuter commande
8. ssh_channel_read() - Lire résultats
9. ssh_channel_close() - Fermer le canal
10. ssh_disconnect() - Fermer la connexion

Commandes distantes exécutées :

```
ps aux --no-headers | awk '{print $1,$2,$3,$4,$8,$11}'
```

Cette commande fournit : - USER : utilisateur - PID : identifiant processus - %CPU : utilisation CPU - %MEM : utilisation mémoire - STAT : état - COMMAND : nom du processus

4. Implémentation

4.1 Outils et technologies

Composant	Technologie	Version
Langage	C	C99
Interface	ncurses	5.9+
Réseau	libssh	0.9+
Compilateur	gcc	9.0+
Build system	GNU Make	4.0+
Contrôle de version	git	2.0+

4.2 Structure du projet

```
lp25-projet/
├── Makefile              # Fichier de compilation
├── README.md             # Documentation utilisateur
├── .gitignore            # Fichiers à ignorer par git
├── doc/                  # Documentation
|   ├── rapport.md        # Ce rapport
|   └── retex.md          # Retour d'expérience
└── src/                  # Code source
    ├── main.c            # Point d'entrée
    ├── manager.c/h       # Orchestration
    ├── process.c/h       # Gestion processus
    ├── network.c/h       # Connexions SSH
    └── ui.c/h            # Interface ncurses
```

4.3 Makefile

Notre Makefile gère : - **Compilation** : `make` ou `make all` - **Nettoyage** : `make clean` (fichiers objets), `make fclean` (exécutable) - **Recompilation complète** : `make re` - **Flags de compilation** : - `-Wall -Wextra -Werror` : tous les avertissements en erreurs - `-O2` : optimisation niveau 2 - `-g` : symboles de débogage (en mode debug)

Extrait du Makefile :

```
CC = gcc
CFLAGS = -Wall -Wextra -O2
LDFLAGS = -lncurses -lssh

SRCS = src/main.c src/manager.c src/process.c src/network.c src/ui.c
OBJS = $(SRCS:.c=.o)
TARGET = my_htop

all: $(TARGET)

$(TARGET): $(OBJS)
    $(CC) $(OBJS) -o $(TARGET) $(LDFLAGS)

clean:
    rm -f $(OBJS)

fclean: clean
    rm -f $(TARGET)
```

4.4 Gestion de la mémoire

Pour éviter les fuites mémoire, nous appliquons les règles suivantes : - Tout `malloc()` a son `free()` correspondant - Libération des structures en cas d'erreur - Utilisation de `valgrind` pour détecter les fuites

Exemple :

```
Process *processes = malloc(sizeof(Process) * max_processes);
if (!processes) {
    perror("malloc failed");
    return NULL;
}
// ... utilisation ...
free(processes);
```

4.5 Gestion des erreurs

Chaque fonction critique retourne un code d'erreur ou un pointeur NULL en cas d'échec :

```
int send_signal_to_process(pid_t pid, int signal) {
    if (kill(pid, signal) == -1) {
        if (errno == EPERM) {
            fprintf(stderr, "Permission denied for PID %d\n", pid);
            return -1;
        }
        perror("kill");
        return -1;
    }
    return 0;
}
```

4.6 Interface ncurses

L'interface est divisée en plusieurs zones :

```
┌──────────────────────────────────────────────┐
| Header: Titre + infos système                |  (3 lignes)
├──────────────────────────────────────────────┤
| Onglets: [Local] [Server1] [Server2]         |  (1 ligne)
├──────────────────────────────────────────────┤
| Colonnes: PID | USER | CPU% | MEM% | STATE | CMD | (1 ligne)
├──────────────────────────────────────────────┤
|                                              |
|          Liste des processus                 |  (variable)
|          (scrollable)                        |
|                                              |
├──────────────────────────────────────────────┤
| Footer: Raccourcis clavier + messages        |  (2 lignes)
└──────────────────────────────────────────────┘
```

Gestion des couleurs :

```
init_pair(1, COLOR_WHITE, COLOR_BLACK);   // Normal
init_pair(2, COLOR_BLACK, COLOR_CYAN);    // Sélection
init_pair(3, COLOR_RED, COLOR_BLACK);     // Erreur
init_pair(4, COLOR_GREEN, COLOR_BLACK);   // Succès
init_pair(5, COLOR_YELLOW, COLOR_BLACK);  // Avertissement
```

4.7 Configuration réseau

Le fichier de configuration `.config` utilise le format suivant :

```
nom:ip:port:user:pass:type
```

Exemple :

```
server1:192.168.1.100:22:admin:password123:ssh
server2:192.168.1.101:22:root:secretpass:ssh
```

Sécurité : Le fichier doit avoir les permissions `600` (lecture/écriture pour le propriétaire uniquement) :

```
chmod 600 .config
```

Le programme vérifie les permissions au démarrage et refuse de continuer si elles sont trop permissives.

---

5. Tests et validation

5.1 Stratégie de test

Nous avons adopté une approche de test en trois niveaux : 1. **Tests unitaires** : validation de chaque fonction individuellement 2. **Tests d'intégration** : validation de la communication entre modules 3. **Tests système** : validation du comportement global

5.2 Tests fonctionnels

5.2.1 Mode local

Test	Procédure	Résultat attendu	Statut
Affichage processus	Lancer `./my_htop`	Liste des processus affichée	✓
Navigation	Utiliser ↑↓	Sélection se déplace	✓
Recherche	Appuyer sur F4, saisir nom	Processus filtré	✓
Pause processus	Sélectionner, F5	État passe à T (Stopped)	✓
Reprise processus	Sélectionner processus pausé, F8	État revient à R/S	✓
Kill processus	Sélectionner, F7, confirmer	Processus disparu	✓
Permissions	Tuer processus root sans sudo	Erreur affichée	✓

5.2.2 Mode réseau

Test	Procédure	Résultat attendu	Statut
Connexion SSH	`./my_htop -s 192.168.1.100`	Connexion établie	✓
Multi-hôtes	`./my_htop -c .config`	Plusieurs onglets	✓
Navigation onglets	F2/F3	Changement d'onglet	✓
Processus distants	Afficher onglet distant	Liste processus distant	✓
Kill distant	Tuer processus sur machine distante	Processus tué	✓
Erreur connexion	IP invalide	Message d'erreur	✓
Timeout	Serveur non accessible	Message timeout	✓

5.3 Tests non fonctionnels

5.3.1 Performance

Métrique	Valeur mesurée	Objectif	Statut
Rafraîchissement local	< 0.5s	< 1s	✓
Rafraîchissement réseau	< 2s	< 3s	✓
Utilisation mémoire	~5MB	< 50MB	✓
Utilisation CPU	~2%	< 10%	✓

Test de charge : - Système avec 2000+ processus : aucune dégradation visible - 5 connexions SSH simultanées : fluide

5.3.2 Sécurité

Test	Résultat
Fichier .config avec permissions 644	Programme refuse de démarrer
Mot de passe dans historique bash	Aucun mot de passe visible
Injection commande SSH	Échappement correct des caractères spéciaux

5.3.3 Robustesse

Scénario	Comportement
Processus disparu pendant refresh	Géré sans crash
Serveur SSH déconnecté	Message d'erreur, onglet désactivé
Redimensionnement terminal	Interface redimensionnée automatiquement
Ctrl+C / SIGINT	Nettoyage propre, ncurses restauré

5.4 Outils de test

5.4.1 Valgrind

Détection des fuites mémoire :

```
valgrind --leak-check=full --show-leak-kinds=all ./my_htop
```

Résultat : Aucune fuite mémoire détectée

5.4.2 GDB

Débogage en cas de crash :

```
gdb ./my_htop
(gdb) run
(gdb) backtrace  # en cas de crash
```

5.4.3 Strace

Analyse des appels système :

```
strace -o trace.log ./my_htop
```

Utile pour vérifier : - Les fichiers /proc ouverts - Les connexions réseau établies

---

6. Guide d'utilisation

6.1 Installation

6.1.1 Dépendances

Debian/Ubuntu :

```
sudo apt-get update
sudo apt-get install build-essential libncurses5-dev libssh-dev
```

CentOS/RHEL :

```
sudo yum groupinstall "Development Tools"
sudo yum install ncurses-devel libssh-devel
```

6.1.2 Compilation

```
cd lp25-projet
make
```

L'exécutable `my_htop` est généré à la racine.

6.2 Utilisation en mode local

6.2.1 Lancement simple

```
./my_htop
```

6.2.2 Avec droits root

Certaines opérations (kill de processus système) nécessitent les droits root :

```
sudo ./my_htop
```

6.2.3 Test d'accès

Pour vérifier les permissions sans lancer l'interface :

```
./my_htop --dry-run
```

6.3 Utilisation en mode réseau

6.3.1 Connexion à un serveur unique

Avec saisie interactive :

```
./my_htop -s 192.168.1.100
```

Le programme demandera le nom d'utilisateur et le mot de passe.

Avec format login :

```
./my_htop -l admin@192.168.1.100
```

Avec tous les paramètres :

```
./my_htop -s 192.168.1.100 -u admin -p password -P 2222
```

6.3.2 Connexion multi-serveurs avec fichier de configuration

Créer le fichier de configuration :

```
nano .config
```

Contenu exemple :

```
production:192.168.1.100:22:admin:pass1:ssh
devel:192.168.1.101:22:dev:pass2:ssh
backup:192.168.1.102:2222:backup:pass3:ssh
```

Sécuriser le fichier :

```
chmod 600 .config
```

Lancer avec le fichier :

```
./my_htop -c .config
```

Inclure aussi le local :

```
./my_htop -c .config -a
```

6.4 Raccourcis clavier

Touche	Action
F1 ou **h**	Afficher l'aide
F2	Onglet précédent (mode réseau)
F3	Onglet suivant (mode réseau)
F4 ou **/**	Rechercher un processus
F5 ou **p**	Pause (SIGSTOP)
F6 ou **k**	Arrêter (SIGTERM)
F7 ou **9**	Tuer (SIGKILL)
F8 ou **c**	Reprendre (SIGCONT)
↑ / **↓**	Naviguer dans la liste
PgUp / **PgDn**	Navigation rapide
Home	Aller au premier processus
End	Aller au dernier processus
q ou **Q**	Quitter
Esc	Annuler (dans les dialogues)

6.5 Options de ligne de commande

Option	Paramètre	Description
`-h`, `--help`	-	Affiche l'aide
`--dry-run`	-	Test l'accès aux processus
`-c`, `--remote-config`	`<file>`	Fichier de configuration serveurs
`-s`, `--remote-server`	`<host>`	Adresse IP/hostname du serveur
`-l`, `--login`	`<user@host>`	Format user@host
`-u`, `--username`	`<user>`	Nom d'utilisateur SSH
`-p`, `--password`	`<pass>`	Mot de passe SSH (non recommandé)
`-P`, `--port`	`<port>`	Port SSH (défaut: 22)
`-t`, `--connexion-type`	`<type>`	Type de connexion (ssh)
`-a`, `--all`	-	Inclure le local avec le distant

6.6 Exemples d'utilisation

Exemple 1 : Surveillance locale simple

```
./my_htop
```

→ Affiche les processus locaux, navigation avec flèches, quitter avec 'q'

Exemple 2 : Tuer un processus spécifique en local

```
./my_htop
# Appuyer sur F4
# Taper "firefox"
# Sélectionner le processus
# Appuyer sur F7 pour SIGKILL
# Confirmer avec 'y'
```

Exemple 3 : Surveiller 3 serveurs de production

```
# Créer .config avec les 3 serveurs
echo "web01:10.0.1.10:22:monitor:pass1:ssh" > .config
echo "web02:10.0.1.11:22:monitor:pass2:ssh" >> .config
echo "db01:10.0.1.20:22:monitor:pass3:ssh" >> .config
chmod 600 .config

# Lancer
./my_htop -c .config

# Naviguer avec F2/F3 entre les onglets
```

Exemple 4 : Surveillance combinée local + distant

```
./my_htop -c .config -a
# Onglet 1: Local
# Onglet 2+: Serveurs distants
```

7. Conclusion

7.1 Résultats obtenus

Le projet a abouti à un outil fonctionnel répondant à l'ensemble des exigences du cahier des charges :

Fonctionnalités implémentées : - ✓ Mode local avec affichage temps réel des processus - ✓ Calcul des métriques CPU et mémoire - ✓ Gestion des processus via signaux UNIX - ✓ Interface ncurses avec navigation intuitive - ✓ Recherche et filtrage de processus - ✓ Mode réseau avec support multi-machines - ✓ Connexions SSH sécurisées - ✓ Système d'onglets pour navigation entre hôtes - ✓ Gestion des erreurs robuste - ✓ Documentation complète

7.2 Limites actuelles

Certaines améliorations pourraient être apportées : - Support des clés SSH (actuellement seul le mot de passe est supporté) - Tri dynamique par colonnes (CPU, mémoire, nom, etc.) - Graphiques d'utilisation des ressources - Export des données (CSV, JSON) - Configuration personnalisable des couleurs - Mode démon pour surveillance continue

7.3 Compétences développées

Ce projet nous a permis de développer de nombreuses compétences techniques et méthodologiques :

Compétences techniques : - Maîtrise de la programmation système Linux - Expertise en langage C et gestion mémoire - Utilisation avancée de ncurses - Implémentation du protocole SSH - Débogage et optimisation

Compétences méthodologiques : - Conception d'architecture logicielle - Gestion de projet informatique - Documentation technique - Travail collaboratif avec git - Tests et validation

7.4 Perspectives

Ce projet constitue une base solide pour d'éventuelles évolutions futures : - **Extension fonctionnelle** : ajout de graphiques, alertes, logs - **Portabilité** : support d'autres systèmes UNIX (BSD, macOS) - **Interface** : version GUI avec GTK ou Qt - **Architecture** : version client-serveur avec agent distant dédié - **Protocoles** : support SNMP pour équipements réseau

7.5 Appréciation personnelle

Ce projet a été extrêmement formateur et nous a permis de comprendre en profondeur le fonctionnement d'un système Linux. Malgré les difficultés rencontrées, notamment lors de la prise en main de ncurses et de l'implémentation du mode réseau, nous sommes fiers du résultat obtenu.

L'expérience acquise sur la gestion des processus, la communication réseau et le développement d'interfaces utilisateur en terminal nous sera très utile dans notre future carrière professionnelle.

---

8. Annexes

8.1 Exemples de sorties

8.1.1 Mode local

```
My HTOP v1.0 - Local Mode                    Load Avg: 0.45, 0.52, 0.48
Memory: 4.2GB / 16GB (26%)                   CPU: 12.5%

  PID    USER      CPU%   MEM%   STATE   COMMAND
──────────────────────────────────────────────────────────────
  1234   abir      25.3   2.1    R       firefox
  5678   abir      12.1   1.5    S       code
  9012   root      0.3    0.1    S       sshd
  3456   abir      0.1    0.5    S       bash

F1:Help F4:Search F5:Pause F6:Kill F7:Force F8:Resume Q:Quit
```

8.1.2 Mode réseau

```
My HTOP v1.0 - Network Mode

[Local] [Server1: 192.168.1.100] [Server2: 192.168.1.101]

Server1 (192.168.1.100) - Connected
  PID    USER      CPU%   MEM%   STATE   COMMAND
──────────────────────────────────────────────────────────────
  1001   www-data  45.2   8.3    R       nginx
  1002   mysql     12.5   15.2   S       mysqld
  1003   root      0.5    0.2    S       sshd

F2/F3:Switch Tab F4:Search F5:Pause F6:Kill Q:Quit
```

8.2 Format des fichiers /proc

/proc/[pid]/stat

```
1234 (firefox) R 1000 1234 1000 ... (52 champs)
```

Champs utilisés : - Champ 1 : PID - Champ 2 : Nom (entre parenthèses) - Champ 3 : État (R/S/D/Z/T) - Champ 14 : utime (temps utilisateur) - Champ 15 : stime (temps système)

/proc/[pid]/status

```
Name:   firefox
State:  R (running)
Uid:    1000    1000    1000    1000
VmSize: 2048576 kB
VmRSS:  512000 kB
```

8.3 Codes d'état des processus

Code	État	Description
R	Running	En cours d'exécution
S	Sleeping	En attente (interruptible)
D	Disk Sleep	En attente (non interruptible, E/S)
Z	Zombie	Terminé mais non collecté
T	Stopped	Arrêté (SIGSTOP)
t	Tracing Stop	En cours de traçage
X	Dead	Mort (ne devrait pas apparaître)

8.4 Signaux UNIX utilisés

Signal	Valeur	Description	Peut être ignoré
SIGSTOP	19	Pause le processus	Non
SIGCONT	18	Reprend le processus	Non
SIGTERM	15	Demande arrêt propre	Oui
SIGKILL	9	Terminaison forcée	Non

8.5 Bibliographie et références

- **The Linux Programming Interface** - Michael Kerrisk (2010)
- **Advanced Programming in the UNIX Environment** - Stevens & Rago (2013)
- **Documentation ncurses** : https://invisible-island.net/ncurses/
- **Documentation libssh** : https://www.libssh.org/
- **Linux /proc filesystem** : https://www.kernel.org/doc/html/latest/filesystems/proc.html

- **Man pages Linux :** `man proc`, `man kill`, `man signal`

8.6 Licence

Ce projet est réalisé dans un cadre pédagogique. Le code est fourni « tel quel », sans garantie d'aucune sorte.

Date de finalisation : Janvier 2026

Signatures :

ISLAM Abir
MELLOUK Mohamed-Amine
FALLANI Issam

Université de Technologie de Belfort-Montbéliard (UTBM)
Licence Professionnelle 25