GLO-2001, Hiver 2022  
Travail Pratique 2

<Votre Nom> - <Votre IDUL>

# Niveau de priorité des *threads* dans Linux

## Programmation de *threads* avec niveaux de priorité

|  |
| --- |
| Sortie de $ ./priority pour le cas a) |
|  |

|  |
| --- |
| Sortie de $ ./priority pour le cas b) |
|  |

|  |
| --- |
| Sortie de $ ./priority pour le cas c) |
|  |

|  |
| --- |
| Sortie de $ ./priority pour le cas d) |
|  |

|  |
| --- |
| Sortie de $ ./priority pour le cas e) |
|  |

f) Explications du résultat:

Pour les cas a. b et c, on remarque que setpriority retourne 0 donc success. Setpriority met la variable globale errno à 0 également. Pour le cas d, par défaut à la création de chaque thread, la priorité est de 0 par conséquent le thread n’a pas le privilège de se mettre une priorité plus basse. Raison pour laquelle setpriority retourne -1 et errno = 13 indique un code d’erreur **PERMISSION DENIED.**

Dans le cas e, on démarre le code en mode administrateur, donc les threads ont la possibilité d’effectuer tous les changements.

## Expérimentations avec différents algorithmes d’ordonnancement

### Sortie du programme

|  |
| --- |
| Sortie de $ ./priority pour le cas FIFO |
| Une image contenant texte  Description générée automatiquement |

|  |
| --- |
| Sortie de $ ./priority pour le cas Round Robin |
| Une image contenant texte  Description générée automatiquement |

# Explications du résultat:

# Une image contenant texte, tableau de points Description générée automatiquement

# Une image contenant texte Description générée automatiquement

# La différence entre les deux sorties ./priority se voit en utilisant la commande top -H dans le terminal. Dans la 1ere image (cas du FIFO) on voit bien qu’il n’y a qu’un seul processus en exécution qui utilise la totalité du cpu. Dans la 2e c’est le Round Robin qui est utilisé. Il y’a partage équitable entre tous les processus.

# Problème de producteurs consommateurs

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de producteurs | Nombre de consommateurs | Nombre d’items | Ce cas fonctionne |
| 3 | 3 | 50 |  |
| 10 | 5 | 100 |  |
| 5 | 10 | 100 |  |
| 20 | 20 | 500 |  |

|  |
| --- |
| Sortie de $ ./file pour le cas 3 producteurs, 3 consommateurs, 50 items. |
|  |