TD de stème n° : arcours de ré ertoires – fork, ait, exit

Dan la uite, on utili era le fonction uivante :

`pi _t fork(voi );`

- crée un clone du proce u appelant duplique tout l'e pace d'adre age ;
- renvoie : le PID du fil dan le proce u père, 0 dan le proce u fil ;
- renvoie -1 en ca d'erreur.

`pi _t wait(int *status);`

- attend la terminai on d'un proce u fil ;
- renvoie le PID du fil qui 'e t terminé ou -1 en ca d'erreur ;
- quand `wait` e débloque `status` contient la valeur du code de retour du fil .

`voi xit(int status);`

- termine l'exécution d'un proce u et demande au y tème de le upprimer ;
- la valeur `status` e t pa ée au père 'il e t en attente ur un `wait`.

Ex rcic 1 : Un r nanc ur

- Écrire une fonction `voi or onanc ur1(int n)`, qui devra ordonancer `n` tâche uppoément $0, \ldots, n-1$. Pour exécuter une tâche, on utili era la fonction uppo ée exi ter `char x cut (int tach )` qui renvoie le lap de temp en econde , inférieur à 4 minute que la tâche `tach` tient à voir 'écouler avant d'être réexécutée. n di po e de la fonction `int att n r (int ur )` qui attend `ur` econde et renvoie `ur` . Notre ordonanceur commencera par lancer toute le tâche imultanément.
- ette foi , au lieu de renvoyer un lap de temp , `x cut` renvoie l'heure à laquelle la tâche veut e réexécuter. n di po e de la fonction `int sonn ()` qui " onne l'heure" : elle renvoie, à l'heure pile, l'heure qu'il e t. Implémentez `voi or onanc ur2(int n)`. Si une tâche termine à 8h35 et demande à être réexécutée à 8h, il faut la relancer in tantanément. Là encore notre ordonanceur commencera par lancer toute le tâche imultanément. n uppo e que votre ordonanceur e t lancé à minuit 0h .

Ex rcic 2 :

Écrire une commande qui prend en argument le chemin d'un répertoire et affiche la tructure de l'arbore cence de fichier à partir du répertoire donné. Exemple d'affichage :

```
 Y TEME
| TD7
  | t 7.p f
  | t 7.t x
| TP7
  | tp7.c
  | tp7.p f
  | tp7.t x
```