18- Par quelle option on doit changer la variable i pour avoir un affichage cohérant :
pthread_create (&tid [i], NULL, &fonction, (void *) &i)

a- (void*)(intptr_t) i
b- (void*) i
c- (int*)i

19- Qu'affiche ce code :

```
void *my_thread_process (void * arg)
{
  for (int i = 0 ; i < 3 ; i++) {
  printf ("Thread %s: %d |", (char*)arg, i);
}
 pthread_exit (0);
}

main (int ac, char **av)
{
pthread_t th1, th2;
   (pthread_create (&th1, NULL, my_thread_process, "1")
   (pthread_create (&th2, NULL, my_thread_process, "2")
pthread_join (th1, NULL);
pthread_join (th2, NULL);

}
```

a- Thread 1 : 0 | Thread 2 : 0 | Thread 1 : 1 | Thread 2 : 1 | Thread 1 : 2 | Thread 2 : 2
b- Thread 0 : 0 | Thread 1 : 0 | Thread 0 : 1 | Thread 1 : 1 | Thread 0 : 2 | Thread 1 : 2
c- Thread 1 : 1 | Thread 2 : 1 | Thread 1 : 2 | Thread 2 : 2 | Thread 1 : 3 | Thread 2 : 3

20- Quel est le type de retour de la fonction pthread_create() :

a- int
b- void
c- void*
d- char*

Institut Supérieur d'Informatique et de Mathématiques de Monastir ISIMM

Examen de rattrapage – S2 – 2023/2024

Filière : *1ère LInfo*	Matière : Système d'exploitation 2		Enseignant : Sana BENZARTI
Date : 10 / 06 / 2024	Nbr de Crédits : 4	Coefficient : 2	Documents autorisés : Non
Durée de l'examen : 1h30	Régime d'évaluation : Mixte / CC		Nombre de pages : 04
	EX (70%) + DS (10%) + TP (20%)		

Choisir la ou les bonnes réponses : barème approximatif (1pt X 20)

1- La syntaxe du Passage d'arguments avec exec:

a- execlp(argv[1], argv[1], NULL)
b- execlp(argv[1], &argv[1], NULL)
c- execv(&argv[1], argv[1])
d- execvp(argv[1], &argv[1])

2- La commande execl permet de passer des arguments de type :

a- Vecteur
b- Liste
c- Entiers
d- Chaine de caractères

3- WIFEXITED(status) et WEXITSTATUS(status) nous indique

a- Le bon fonctionnement du programme
b- La liste des processus
c- Un signal de terminaison
d- La façon dont le processus s'est terminé

4- pthread_create(&thread1, NULL, fct_th, (void*)&tab) utilise un argument de type

a- Un entier
b- Un tableau d'entiers

5- WIFSIGNALED(status) et WTERMSIG(status) nous indique

a- Un signal extérieur envoyé vers le processus
b- Une erreur dans le code
c- Une fin urgente pour le processus
d- Une terminaison immédiate en utilisant un signal SIGKILL et un code de retour égal a 9
e- Une terminaison propre en utilisant un signal SIGTERM et un code de retour égal a 15

6- WIFEXITED est une macro utilisée :

a- Avec la fonction exit
b- Avec la fonction fork
c- Avec la fonction wait

7- La commande execvp permet de passer des arguments de type :

a- Vecteur
b- Liste
c- Chaine de caractère

8- La fonction pthread_join(thread1, NULL) :

a- Permet d'attendre le thread 1 jusqu'a ce qu'il envoie son code de retour
b- Permet d'attendre n'importe quel thread
c- Permet de mettre fin au thread 1

9- Un interblocage se produit lorsque :

a- Toutes les ressources sont disponibles
b- Il y a un accès concurrent
c- Conflit d'accès au ressources partagées

10- Les mutex permettent de :

a- Synchroniser l'accès vers les ressources partagées
b- Imposer un verrou sur les threads
c- Mettre fin aux threads

11- On peut initialiser un mutex en utilisant la syntaxe suivante :

a- pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER
b- pthread_mutex_init(&mutex, NULL)
c- pthread_mutex_lock(&mutex)

12- Que fait ce code :

```
void *fct(void *arg) {
  pthread_mutex_lock(&mutex);
  printf("thread 1  utilise la variable var 1.\n");
  var1++;
 printf("thread 1  utilise la variable var 2.\n");
```



```
  var2--;
 pthread_mutex_unlock(&mutex);
  pthread_exit(NULL);
}
```

a- Impose le verrouillage des variables var 1 et var 2 jusqu'a ce que thread 1 complète sa fonction
b- Impose le verrouillage de la variable var 1 jusqu'a ce que thread 1 complète sa fonction
c- Impose le verrouillage de la variable var 2 jusqu'a ce que thread 1 complète sa fonction

13- Avec la fonction pthread_detach() :

a- Les ressources seront libérées et le thread principal n'attend pas le thread secondaire
b- Les ressources ne sont pas libérées et le thread principal n'attend pas le thread secondaire
c- Les ressources ne sont pas libérées et le thread principal attend le thread secondaire

14- La fonction pthread_self() renvoie :

a- Le PID du thread
b- Le TID du thread

15- Cette syntaxe < char* var = (char*)arg > permet :

a- De convertir un pointeur spécifique vers un pointeur générique
b- De convertir un pointeur générique vers un pointeur spécifique

16- Le heap ou le tas est :

a- Une zone mémoire statiquement allouée à l'exécution pour stocker des données qui sont créées et détruites de manière statique
b- Une zone de mémoire dynamique allouée à l'exécution pour stocker des données qui sont créées et détruites de manière dynamique
c- C'est une pile

17- Dans le cycle de vie d'un thread, on peut trouver les états suivants :

a- Prêt → actif → bloqué → actif → terminé
b- Prêt → actif → terminé
c- Prêt → nettoyé
d- Prêt → actif → bloqué → prêt → actif → terminé