Commencé le	dimanche 28 janvier 2024, 17:27
État	Terminé
Terminé le	dimanche 28 janvier 2024, 17:30
Temps mis	3 min 11 s
Note	**20,00** sur 20,00 (**100**%)

Question 1

Correct

Note de 1,00 sur 1,00

Un processus peut exécuter plusieurs programmes à la fois.

Veuillez choisir une réponse.

- Vrai
- Faux ✔

La réponse correcte est « Faux ».

Question 2

Correct

Note de 1,00 sur 1,00

Un même programme peut être exécuté par plusieurs processus.

Veuillez choisir une réponse.

- Vrai ✔
- Faux

La réponse correcte est « Vrai ».

Question 3

Correct

Note de 1,00 sur 1,00

Est-ce qu'un processus peut remplacer son programme en cours par un autre ?

Veuillez choisir une réponse.

a. Non.

b. Oui. ✔

Votre réponse est correcte.

La réponse correcte est :
Oui.

Question 4

Correct

Note de 1,00 sur 1,00

Sélectionnez les appels système qui peuvent faire basculer l'état du processus appelant vers l"état bloqué.

Veuillez choisir au moins une réponse.

a. read(0,buf,nb); ✔

b. sleep(5); ✔ le processus passe à l"état bloqué pendant au plus 5 secondes.

c. _exit(0);

d. getpid();

e. wait(NULL); ✔ Le processus appelant passe à l'état bloqué s'il a au moins fils non zombie.

f. sched_yield(); // céder le processeur

Votre réponse est correcte.

Les réponses correctes sont :
read(0,buf,nb);,

sleep(5);,

wait(NULL);

Question 5

Correct

Note de 1,00 sur 1,00

Dans les systèmes d'exploitation de la famille UNIX, en plus des trois états principaux (prêt, en exécution et bloqué), il y a l'état zombie. Sélectionnez les énoncés qui sont vrais.

Veuillez choisir au moins une réponse.

- [x] a. Un processus passe à l'état zombie lorsqu'il se termine. ✔
- [ ] b. Un processus passe à l'état zombie lorsqu'il perd son père.
- [ ] c. Un processus à l'état zombie peut basculer vers l'état prêt.

Votre réponse est correcte.

La réponse correcte est :
Un processus passe à l'état zombie lorsqu'il se termine.

Question 6

Correct

Note de 1,00 sur 1,00

L'état zombie sert à conserver les informations concernant la terminaison d'un processus. Le processus parent peut récupérer, via les appels système wait et waitpid, ces informations.

Veuillez choisir une réponse.

- (•) Vrai ✔
- ( ) Faux

La réponse correcte est « Vrai ».

Question 7

Correct

Note de 1,00 sur 1,00

Le principe de "Copy-On-Write" (COW) permet aux processus de partager des ressources tant qu'ils y accèdent en lecture et d'en créer des copies privées lorsqu'ils y accèdent en écriture.

Veuillez choisir une réponse.

- (•) Vrai ✔
- ( ) Faux

La réponse correcte est « Vrai ».

Question 8

Correct

Note de 1,00 sur 1,00

// https://onlinegdb.com/r12sNarJd

Considérez le programme suivant :

```
int main () {
  pid_t p = fork ();
  printf (" 1er printf de pid=%d\n", getpid ());
  switch (p) {
    case 0:  printf (" 2ieme printf de pid=%d\n", getpid ());
             break;
    case -1: printf (" 3ieme printf de pid=%d\n", getpid ());
              break;
    default: printf (" 4ieme printf de pid=%d\n", getpid ());
              wait (NULL);
    }
  printf (" 5ieme printf de pid=%d\n", getpid ());
  return 0;
  printf (" 6ieme printf de pid=%d\n", getpid ());
}
```

Sélectionnez les instructions *printf* exécutées par le processus fils créé par l'appel système *fork*.

Veuillez choisir au moins une réponse.

- a. *printf(" 1er printf de pid=%d\n",getpid());* ✔
- b. *printf(" 2ieme printf de pid=%d\n",getpid());* ✔
- c. *printf(" 3ieme printf de pid=%d\n",getpid());*
- d. *printf(" 4ieme printf de pid=%d\n",getpid());*
- e. *printf(" 5ieme printf de pid=%d\n",getpid());* ✔
- f. *printf (" 6ieme printf de pid=%d\n", getpid ());*

Votre réponse est correcte.

Les réponses correctes sont :
printf(" 1er printf de pid=%d\n",getpid());,

printf(" 2ieme printf de pid=%d\n",getpid());,

printf(" 5ieme printf de pid=%d\n",getpid());

Question 9

Correct

Note de 1,00 sur 1,00

Considérez le programme suivant :

// https://onlinegdb.com/IWpNIjjSq

```
int main () {
  if (fork () == 0) {
     printf ("avant execl : processus de pid=%d de père %d\n", getpid (), getppid ());
     execl ("/bin/ls", "ls", "-l", NULL);
     printf ("après execl : processus de pid=%d de père %d\n", getpid (), getppid ());
   }
  int f=wait (NULL);
  printf ("fin du fils %d\n", f);
  return 0;
}
```

Sélectionnez toutes les instructions exécutées par le processus fils, dans le cas où l'exécution du programme se déroule sans erreurs.

Veuillez choisir au moins une réponse.

- [x] a. *printf("avant execl : processus de pid=%d de père %d\n",getpid(), getppid());* ✔
- [x] b. *execl("/bin/ls", "ls", "-l", NULL);* ✔
- [ ] c. *f = wait(NULL);*
- [ ] d. printf("après execl : processus de pid=%d de père %d\n",getpid(), getppid());

Votre réponse est correcte.

Les réponses correctes sont :
printf("avant execl : processus de pid=%d de père %d\n",getpid(), getppid());,

execl("/bin/ls", "ls", "-l", NULL);

Question 10

Correct

Note de 1,00 sur 1,00

Considérez le programme suivant :

```
//https://onlinegdb.com/r1_7lOLyu
int main () {
  if (fork () == 0) {
     printf ("avant execl : pid=%d\n", getpid ());
     execl ("ls", "ls", "-l", NULL);
     printf ("après execl : pid=%d %d\n", getpid ());
   }
  int f=wait (NULL);
  printf ("fin du fils %d\n", f);
  return 0;
}
```

Sélectionnez chaque **instruction exécutée par le processus père et aussi par le processus fils**, dans le cas où l'appel système ***execl* échoue**.

Veuillez choisir au moins une réponse.

- a. *printf ("avant execl : pid=%d\n", getpid ());*
- b. *printf ("après execl : pid=%d %d\n", getpid ());*
- c. *f=wait (NULL);* ✔
- d. *printf ("fin du fils %d\n", f);* ✔
- e. *return 0;* ✔
- f. *execl ("ls", "ls", "-l", NULL);*

Votre réponse est correcte.

Les réponses correctes sont :
f=wait (NULL);,

printf ("fin du fils %d\n", f);,

return 0;

Question 11

Correct

Note de 1,00 sur 1,00

L'appel système fork permet à un processus de créer un processus fils. Un processus crée, via fork, un processus fils. Le processus fils créé partage avec son père ...

Veuillez choisir au moins une réponse.

- [x] a. les pages du père présentes en mémoire physique aussi longtemps que le père et le fils y accèdent en lecture. ✔
- [ ] b. le pointeur de lecture/ écriture de tout fichier ouvert par son père après le fork.
- [x] c. le pointeur de lecture/ écriture de tout fichier ouvert par son père avant le fork. ✔
- [ ] d. Aucune réponse

Votre réponse est correcte.

Les réponses correctes sont :
les pages du père présentes en mémoire physique aussi longtemps que le père et le fils y accèdent en lecture., le pointeur de lecture/ écriture de tout fichier ouvert par son père avant le fork.

Question 12

Correct

Note de 1,00 sur 1,00

Considérez le code suivant :

```
int main() {
   int i=0 ;
   if(fork()) {i=i+1;}
   i=i+1;
  printf("i=%d \n", i);
  wait(NULL);
  return 0;
}
```

Les valeurs de i affichées par le processus principal PP et le fils F de PP sont :

Veuillez choisir une réponse.

- (●) a. PP:i=2 et F:i=1. ✔
- ( ) b. PP:i=1 et F:i=2.
- ( ) c. PP:i=2 et F:i=2.
- ( ) d. aucune péponse

Votre réponse est correcte.

La réponse correcte est :
PP:i=2 et F:i=1.

Question 13

Correct

Note de 1,00 sur 1,00

Un programme crée trois processus qui exécutent la même fonction MyCode() puis se terminent. Cette fonction consiste en un traitement local et une incrémentation de 1 d'un compteur (déclaré avant la fonction main). Le programme attend ensuite la fin des trois processus, affiche la valeur du compteur puis se termine. La valeur initiale du compteur est 0.

Le programme est-il déterministe (il affichera toujours la même valeur) ?

Veuillez choisir une réponse.

a. Oui, car il affichera toujours la valeur 3.

b. Non, car il affichera 1, 2 ou 3.

c. Non, car il affichera 0, 1, 2 ou 3.

d. Oui, car il affichera toujours 0. ✔

e. Aucune réponse

Votre réponse est correcte.

Toute modification de la variable compteur réalisée par un processus sera invisible par les autres processus.

La réponse correcte est :
Oui, car il affichera toujours 0.

Question 14

Correct

Note de 1,00 sur 1,00

Un processus exécute le code suivant :

```
int x = 100;
int main ()
{  if (fork () > 0) {
        x = x + 1;
        wait (NULL);
        printf ("%d\n", x);
   }  else {
         x = x + 2;
   }
  _exit (0);
}
```

La valeur de x affichée par « printf » est :

Veuillez choisir une réponse.

- a. 101. ✔
- b. 103.
- c. 102.
- d. 101 ou 103.
- e. aucune réponse.

Votre réponse est correcte.

La réponse correcte est :
101.

Question 15

Correct

Note de 2,00 sur 2,00

Quel est le nombre de processus créés par la séquence d'instructions suivante, si l'exécution se déroule sans erreurs ?

```
fork();
fork();
execlp("ls", "ls", NULL) ;
fork();
fork();
```

Veuillez choisir une réponse.

- a. 2.
- b. 3. ✔
- c. 4.
- d. 6.
- e. 8.
- f. aucune réponse.

Votre réponse est correcte.

Le processus principal PP (celui qui commence ce code) va réaliser 2 appels à la fonction fork puis se transforme en ls. Il va donc créer 2 processus fils F1 et F2. Le premier fils F1 va réaliser 1 appel à la fonction fork avant de se transformer en ls. Les autres processus créés vont se transformer en ls.

Au total, ce code crée 3 processus.

La réponse correcte est :
3.

Question 16

Correct

Note de 2,00 sur 2,00

Considérez le code suivant :

```
// https://onlinegdb.com/1QaKLp_eDB
int main () {
  int p , n = 0;
  while ( (p=fork()) > 0)   n++;
  if (p == 0)  sleep (1);
  else  while (wait (NULL) > 0);
  return 0;
}
```

Que fait ce programme ?

Veuillez choisir au moins une réponse.

- [x] a. Il va créer des processus à répétition jusqu'à ce que l'appel à fork retourne -1 ✔
- [x] b. À la sortie de la boucle, le processus principal récupére dans la variable n le nombre de processus qu"il a créés. ✔
- [ ] c. À la sortie de la boucle, la valeur de p est toujours le pid du dernier processus créé.
- [x] d. Chaque processus fils créé va dormir pendant 1 seconde avant de se terminer. ✔
- [ ] e. À la sortie de la boucle, la valeur de n est toujours 1.
- [ ] f. Il va créer un processus fils, attendre la fin de son fils puis se termine.

Votre réponse est correcte.

Le processus principal va créer des processus à répétition jusqu'à ce que l'appel à fork retourne -1. Il va attendre ensuite la fin de tous ses fils. À la sortie de la boucle, il récupérera dans la variable n le nombre de processus créés.

Chaque processus fils dort pendant 1 seconde puis se termine.

Les réponses correctes sont :
Il va créer des processus à répétition jusqu'à ce que l'appel à fork retourne -1,

À la sortie de la boucle, le processus principal récupére dans la variable n le nombre de processus qu"il a créés.,

Chaque processus fils créé va dormir pendant 1 seconde avant de se terminer.

Question 17

Correct

Note de 1,00 sur 1,00

Un descripteur de fichier est un nombre entier qui :

Veuillez choisir au moins une réponse.

- a. est associé à toutes les ouvertures d'un même fichier.
- b. identifie, au niveau d'un processus, un fichier ouvert. ✔
- c. est identique à l'i-noeud d'un fichier ouvert.
- d. permet d'accéder, à partir d'un processus, à un fichier ouvert. ✔

Votre réponse est correcte.

Les réponses correctes sont :
identifie, au niveau d'un processus, un fichier ouvert.,

permet d'accéder, à partir d'un processus, à un fichier ouvert.

Question 18

Correct

Note de 1,00 sur 1,00

```
// programme PrintfFork2.c
#include <sys/wait.h> // pour wait
#include <stdio.h>
#include <unistd.h>
int main() {
      printf("bonjour je suis le processus principal");
      int cpid = fork();
     if (!cpid) {
         printf("Ici le fils!\n");
        _exit(0);
     }
    wait(NULL);
   printf("Fin du programme\n");
   return 0;
}
```

Ce programme va afficher ce qui suit :

Veuillez choisir une réponse.

a. bonjour je suis le processus principalIci le fils! ✔
bonjour je suis le processus principalFin du programme
bonjour je suis le processus

b. bonjour je suis le processus principalIci le fils!
Fin du programme

c. bonjour je suis le processus principalFin du programme
bonjour je suis le processus principalIci le fils!

d. aucune réponse

Votre réponse est correcte.

L'appel à fork crée un processus par duplication selon le principe copy-on-write. Le contenu du buffer associé à la sortie standard est aussi dupliqué. Le fils créé par l'appel à fork va donc hériter du contenu de ce buffer.

La réponse correcte est :
bonjour je suis le processus principalIci le fils!

bonjour je suis le processus principalFin du programme