 NOM : Prénom :

Devoir sur Table : Programmation Système

Enseignant : Anthony Busson

Exercice 1 :

Nous considérons le code ci-dessous. Donnez le résultat de l'affichage de ce programme. Vous pourrez donner les pid que vous souhaitez, il faut juste qu'ils soient cohérents vis-à-vis de l'exécution du programme.

```
int main()
{
	int i, retourExit, retourFork=0, pid;

	for(i=0; i < 4; i++)
	{
		if(retourFork==0)
		{
			printf("Mon PID est %d, i=%d, et mon père a le PID=%d\n",getpid(),i,getppid());
			retourFork=fork();
		}
	}
	if( (pid=wait(&retourExit))>0 ) printf("Code de retour du fils %d: %d\n",pid,WEXITSTATUS(retourExit));
	exit(i);
}//fin du main
```

Exercice 2 :

L'exercice consiste à écrire le code permettant de manipuler un fichier en format brut. Vous devez compléter le code ci-dessous à 3 endroits numéroté 1, 2 et 3. Pour les points 1 et 2 vous devez mettre la structure à jour. Pour le point 3, vous devez placer la structure uneBlague en quatrième position dans le fichier. Le format de la structure devra être conservé. On suppose que le fichier contient déjà 10 blagues (10 occurrences de la structure toto).

Le code :

```
struct toto{
        int numero ;
        char blague[100] ;
}
int main()
{
        int fd ;
        struct toto uneBlague;

        //1. Mettre à jour le numéro de la structure uneBlague avec le numéro 4


        //2. Mettre à jour la chaine de caractère de la structure toto avec "C'est toto qui …"


        if(  (fd=open(« fichier », O_WRONLY))<0) perror("Erreur open");
        //3. Il faut placer la structure uneBlague à la quatrième position dans le fichier




        close(fd);
        return(0);
}//fin du main
```

Manuels:

lseek - Positionner la tête de lecture/écriture dans un fichier

off_t lseek(int *fd*, **off_t** *offset*, **int** *whence*);

DESCRIPTION

La fonction **lseek**() place la tête de lecture/écriture à la position *offset* dans le fichier ouvert associé au descripteur *fd* en suivant la directive *whence* ainsi :
SEEK_SET La tête est placée à *offset* octets depuis le début du fichier.
SEEK_CUR La tête de lecture/écriture est avancée de *offset* octets.
SEEK_END La tête est placée à la fin du fichier plus *offset* octets.

La fonction **lseek**() permet de placer la tête au-delà de la fin actuelle du fichier (mais cela ne modifie pas la taille du fichier). Si des données sont écrites à cet emplacement, une lecture ultérieure de l'espace intermédiaire (un « trou ») retournera des zéros (« \0 ») jusqu'à ce que d'autres données y soient écrites.

VALEUR RENVOYÉE
lseek(), s'il réussit, renvoie le nouvel emplacement, mesuré en octets depuis le début du fichier. En cas d'échec, la valeur *(off_t) -1* est renvoyée, et *errno* contient le code d'erreur.

Exercice 3:

Nous considérons le bout de code ci-dessous. Vous devrez décrire le contenu du tableau des fichiers ouverts (voir plus bas).

```
int main()
{
        int fd1, fd2 ;
        if(  (fd1=open(« fichier1.txt », O_WRONLY|O_CREAT,0666))<0) perror(“Erreur open
        fichier1.txt”);
        if(  (fd2=open(« fichier2.txt », O_WRONLY|O_CREAT,0666))<0) perror(“Erreur open
        fichier2.txt”);
        dup2(fd1,2);
        dup2(2,fd2);
        return(0)
}
```

 NOM : Prénom :

Complétez le tableau ci-dessous de manière à indiquer vers quel fichier pointe chaque case du tableau après la dernière instruction (vers stdin, le fichier1.txt, etc.). Tous les pointeurs de 0 à 5 inclus doivent être renseignés.

Pointeur[0]
Pointeur[1]
Pointeur[2]
Pointeur[3]
Pointeur[4]
Pointeur[5]

Rappel du man de dup et dup2 :

Nom
dup, dup2 - Dupliquer un descripteur de fichier.
Synopsis
#include <unistd.h>
int dup(int oldfd);int dup2(int oldfd, int newfd);
Description
dup et **dup2** créent une copie du descripteur de fichier *oldfd*. Après un appel réussi à **dup** ou **dup2**, l'ancien et le nouveau descripteurs peuvent être utilisés de manière interchangeable. Ils partagent les verrous, les pointeurs de position et les drapeaux. Par exemple si le pointeur de position est modifié en utilisant **lseek** sur l'un des descripteurs, la position est également changée pour l'autre. Les deux descripteurs ne partagent toutefois pas le drapeau Close-on-exec.
dup utilise le plus petit numéro inutilisé pour le nouveau descripteur.
dup2 transforme *newfd* en une copie de *oldfd*, fermant auparavant *newfd* si besoin est.
Valeur Renvoyée
dup et **dup2** renvoient le nouveau descripteur, ou -1 s'ils échouent.

Question de cours :

1. Décrivez les 3 scenarios possibles de filiation entre père et fils (le père meurt d'abord, etc.). Expliquez.

2. Pourquoi certaines commandes du shell sont internes d'autres externes? Que cela signifie-t-il ? Et expliquez la raison.

3. Donnez le prototype de l'appel système execvp() et indiquez à quoi correspondent les arguments ? Que fait cet appel système ? A quoi sert-il ?