Correction examen de révision

Système

Question 1)

- bufferisation interne par la libc c

Un appel système est semblable à un appel de fonction, mais effectue en plus deux changements de niveau de privilège : une transition vers le mode privilégié lors de l'appel, et une transition vers le mode non-privilégié lors du retour. De ce fait, un appel système est lent, typiquement entre 1000 et 10000 fois plus lent qu'un appel de fonction sur du matériel moderne.

- portabilité

- ?

Question 2)

***Les liens physiques :

--> création d'un nouveau nom pour le même inode.
--> impossible de faire un lien physique vers un répertoire : le système de fichiers doit être arborescent.
--> impossible de faire un lien physique vers un autre volume (partition)
-->Quand tu fais un lien physique entre file1 et file2, ils partagent le même inode (identificateur de fichier) donc si tu supprimes file1, file2 reste accessible (et inversement) tu n'as au total qu'un fichier accessible par deux noms différents.

***Les liens symboliques (raccourcis) :

--> création d'un nouvel inode de type lien symbolique
--> le bloc de données contient un chaine de caractères représentant le chemin (relatif ou absolu) vers un fichier.
--> possibilité de lien symbolique vers un repertoire.
--> possibilité de lien symbolique vers un autre volume (partition)
-->Quand tu supprimes la cible, le lien est mort.

Question 3)

Non ; Pas dans les tubes par exemple (tout est fihier sous linux)
"Les tubes sont gérés comme un fichier, dans la table des fichiers ouverts du système. Mais contrairement aux fichiers, aucun transfert (disque ↔ mémoire centrale) n'est effectué et aucun déplacement (lseek()) n'est autorisé."

Question 4)

- il faut le détruire explicitement
- la communication inter-processus

"Les tubes nommés offrent les possibilités générales de communication décrites pour les tubes, en étendant l'accès à des processus appartenant à des utilisateurs différents"*

Question 5)

25 89 1 69 54 13

1: (25 et 13) **13** 89 **1** 69 54 25

2: (13 et 25; 13 et 54; 13 et 69; 13 et 1) **1** 89 **13** 69 54 25

3: (1 avec tout c'est bon on passe a la case suivante)

4: (89 et 25) 1 **25** 13 69 54 **89**

5: (25 et 89; 25 et 54; 25 et 69; 25 et 13) 1 **13** **25** 69 54 89

6: (13 et tout le reste c bon on passe a la prochaine case)

7: (25 et tout l reste c bon on passe a la prochaine case)

8: (69 et 89; 69 et 54) 1 13 25 **54** **69** 89

Question 6) 7)

```
#include <stdio.h>
int main(void)
{
	int i;
	int j;
	int t[] = {25, 89, 1, 69, 54, 13, -345, 32};
	int sz = sizeof(t) / sizeof(int);

	for (i = 0; i < sz; i++)
		for (j = sz - 1; j > i; j--)
			if (t[i] > t[j]) {
				int tmp = t[i];
				t[i] = t[j];
				t[j] = tmp;
			}

	for (i = 0; i < sz; i++)
		printf("%d ", t[i]);
}
```

Question 8)

main =89 1 69 54 13= ps1(1) =89 69 54 13= ps2(13) =89 69 54= ps3(54) =89 69= ps4(69) =89= ps5(89)

Question 9)

25 89 1 69 54 13

P1 stocke premier entier lu **V=13**: **tube in (25 89 1 69 54)**

- 54>V envoyer 54 dans le **tube out (54)** ; **tube in (25 89 1 69)**
- 69>V pareil; **tube out(69 54)**; **tube in (25 89 1)**
- 1<V envoyer V dans le tube out et stocker 1 dans V; **V=1; tube out (13 69 54); tube in (25 89)**
- 89>V envoyer 89 dans tube out; **tube out (89 13 69 54)**; **tube in(25)**
- 25>V pareil; **tube out (25 89 13 69 54); tube in vide;**
- P1 affiche **V=1** et le **tube out (25 89 13 69 54) sera le tube in du P2**

P2 stocke .....

Question 10) 11)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
```

```c
#include <stdlib.h>


void sort(int fd_read)
{
        int buf;
        int min;
        int p[2];
        int s;
        pipe(p);

        if (read(fd_read, &min, sizeof(int))) {
                if (fork()) {
                        close(p[0]);
                        while(read(fd_read, &buf, sizeof(int)))
                                if (min > buf) {
                                        write(p[1], &min, sizeof(int));
                                        min = buf;
                                } else {
                                        write(p[1], &buf, sizeof(int));
                                }
                        close(p[1]);
                        printf("%d\n", min);
                } else {
                        close(p[1]);
                        sort(p[0]);
                }
                wait(&s);
        }
        close(fd_read);
}


int main(int argc, char **argv)
{
        int i;
        int s;
        int buf;
        int p[2];

        pipe(p);

        if (fork()) {
                close(p[0]);
                for (i = 1; i < argc; i++) {
                        buf = atoi(argv[i]);
                        write(p[1], &buf, sizeof(int));
                }
                close(p[1]);
        } else {
                close(p[1]);
                sort(p[0]);
        }

        wait(&s);
}
```