TD3 - ISS

Exercice 1.

1.1

[A INSERER]

1.2

S'on appelle à `/bin/echo` alors n'est plus un built-in et donc on va faire 8 processus.

1.3

En utilisant la commande `type <command>`

1.4

```
 12345: -
67890: --
67890: ----
67890: --------
67890: ----------------
12345: -
```

Le "problème" est que un processus ne partage pas sa mémoire, même si c'est un pêre ou un fils, alors on peut "exporter" la variable.

1.5

Il faut juste utiliser `export` pour ajouter la variable `$chaine` à l'environnement du père (et comme ça, va être là après du fork / exec) ou faire un appel au fils avec `env`.

1.6

L'affichage ne change pas parce que on n'a pas mis en place aucun système de communication dans les processus.

1.7

Le `$PATH`

1.8

La securité et eviter les conflicts de noms.

Exercice 2.

2.1

À inserer.

2.2

C'est possible, on à juste à faire un `echo` après l'appel pelican.sh, comme on n'a pas utilisé `&` le père va à être bloqué jusqu'à la fin de tous ses fils.

2.3

Paramètres:

```
#! /bin/bash
# jonathan.sh arguments

if [[ ${#} -eq 0 ]] ; then
    echo "Usage: jonathan.sh <nombre_executions>"
    exit
fi

./pelican.sh $1
echo "Fin des pelican."
```

```
#! /bin/bash
# pelican.sh arguments

if [[ ${1} -ge 1 ]] ; then
    ./pelican.sh $(($1 - 1))
fi
```

Environnement:

```
#! /bin/bash
# jonathan.sh env

if [[ ${#} -eq 0 ]] ; then
    echo "Usage: jonathan.sh <nombre_executions>"
    exit
fi

export it=$1
./pelican.sh
echo "Fin des pelican."
```

```
#! /bin/bash
# pelican.sh env

if [[ ${it} -ge 1 ]] ; then
    it=$((it - 1))
    ./pelican.sh
fi
```

Summary

exporting variables

If we decide to use `export` to let our process to communicate the exported variable will only be present on the current shell and its child processes.

env vs export

- `env`

`env` allows to spawn a child with a modified system environnement that contains a variable, this only last for that call (and children resultant from that call).

`env VAR=value command`

- `export`

`export` also sets an environnement variable but lasts through all the current shell session and its descendents, after the `export` was made (and not necessarily related to that process)

`export var`

`source`

Communication between fathers and childs

The only way to communicate between father and childs, is using either temporary files, command substitution or Inter-Process Communication.