- D A E M O N L A B -

Préparation

Prise en main du poste
Manipulation du shell

- DaemonLab -

Durant cette journée, vous allez pour la plupart d'entre vous prendre en main un poste sur ***Linux****, équipé en logiciels pour développer. Cet environnement peut vous paraître* ***austère****, vous verrez qu'avec le temps, il s'avérera être un puissant* ***allié****.*

INDEX

01 – Votre poste

02 – L'Infosphère

03 – Rendre votre travail : La méthode NFS

04 – Propreté de votre rendu

05 – La ligne de commande

06 – Liste des fichiers cachés

07 – La configuration de bash

08 – Chaînage d'opérations

01 – Votre poste

Vous faites face à un poste informatique constitué pour sa partie matérielle d'un **Raspberry PI 4**, relié à un service d'authentification **LDAP** et de stockage de fichier **NFS**, et a qui pour sa partie système d'exploitation d'une distribution **GNU/Linux** du nom de **ArchLinux**. L'environnement graphique auquel vous faites face s'appelle **i3**.

Le **Raspberry PI 4** est un micro-ordinateur exploitant un micro-processeur **RISC ARM** Cortex A72 disposant de **quatre** cœurs cadencés à 1,5 Ghz disposant de 4Go de mémoire vive ainsi que d'une carte vidéo permettant l'accélération graphique.

Le service d'authentification **LDAP** et le système réseau de stockage de fichier **NFS** vous permet de vous connecter sur n'importe lequel des postes **Raspberry PI 4** du parc de l'école et d'accéder à vos fichiers.

Linux est le nom du noyau du système d'exploitation : la partie chargée des interactions fondamentales sur le matériel et de la gestion des processus – entre autres.

GNU est le nom d'un ensemble de logiciels réalisant la première couche autour du noyau : les outils qui servent à la construction des programmes, à travailler dans l'environnement de la ligne de commande… Une particularité des logiciels **GNU** est d'être sous **licence GNU** et donc d'être libre.

ArchLinux est le nom d'un ensemble de logiciels et de configuration pré-établies de l'ensemble précédent ainsi que de services en lignes permettant l'adjonction, le retrait et la mise à jour de logiciels sur le système d'exploitation.

L'environnement graphique **i3** est d'un type auquel vous n'avez probablement jamais fait face : il fonctionne sur un principe du découpage de la zone affichable qu'on appelle **mosaïque.** Au lieu d'avoir des fenêtres déplaçable librement sur l'écran, vous pouvez casser l'espace en deux pour créer deux sous-espaces, eux même cassable en deux, etc. Exploitant **bien plus le clavier** et moins la souris.

Actuellement, vous n'y êtes pas habitués, et donc vous devez certainement vous demander à quoi bon…

L'objectif de **i3** n'est pas d'être simple à utiliser, mais d'être extrêmement rapide à l'emploi une fois celui-ci maîtrisé. Vous remarquerez assez vite que c'est la philosophie de beaucoup des logiciels que vous allez apprendre à utiliser. *Vous allez progressivement vous rendre compte que vous utilisez de moins en moins la souris…*

Apprenez à prendre en main votre poste :
https://wiki.lapins-noirs.fr/fr/public/postes
Support vidéo pour la prise en main du poste :
https://video.ecole-89.com/videos/watch/fd1f6319-02d5-4131-8791-09c2d44f3953

Allez plus loin sur i3 :
http://wiki.lapins-noirs.fr/fr/de-i3

02 – L'Infosphère

L'**Infosphère** est un intranet scolaire. Il s'agit d'un logiciel faisant partie d'une suite logicielle sous licence libre s'appelant **TechnoCentre** et dont le propos est d'encadrer de manière totale une scolarité. **Infosphère** peut s'adapter à n'importe quel type d'école mais contient des spécificités supplémentaires utiles à l'apprentissage du développement informatique.

L'**Infosphère** a été pensée pour encadrer **spécifiquement** le projet pédagogique que vous allez vivre en tant qu'étudiant, mais elle n'est pas encore totalement terminée. Elle comporte énormément de fonctionnalités mais pas encore toutes celles qui ont été pensés pour elle, et elle peut encore comporter des bugs. Elle dispose de plus d'une apparence temporaire plutôt austère. Vous aurez donc l'opportunité de voir apparaître régulièrement des nouveautés, des améliorations et si vous souhaitez y contribuer par des idées, des remontées, des critiques, n'hésitez pas. L'**Infosphère** est développée par **Jason Brillante** et l'aspect système du **TechnoCentre** par **Elise Philippe,** qui vous encadrent en ce moment. Les autres logiciels de cette suite sont développés par l'association co-initiatrice de ce logiciel libre, **Pentacle Technologie**.

L'Infosphère contient, entre autres :

1 – Votre calendrier

_ Sont visibles les **activités en salles et les projets**, sur un trimestre entier.

_ Vous pouvez cliquer sur une activité, en salle comme projet, afin d'accéder à une page d'activité comprenant toutes les informations la concernant :

- Son nom, sa description, la taille des équipes…
- Ses dates d'inscription, **de rendu,** d'apparition du sujet…
- **Son sujet**
- Ses encadrants
- Ses inscrits
- Les médailles associées
- Les supports de cours associés et activités pré-requises
- Sur cette page, vous avez la possibilité de :
  - _ Vous **inscrire**
  - _ Choisir une équipe
  - _ Signaler votre présence à l'heure de l'appel

2 – Les modules auquel vous participez, de manière non chronologique

3 – Les supports associés aux activités auxquelles vous participez

4 – **Votre profil, vos médailles, vos crédits, vos évaluations**

5 – D'autres choses que vous découvrirez plus tard.

Les noms **TechnoCentre** et **Infosphère** viennent de l'univers des **Cantos d'Hyperion**. Une série de roman de science-fiction que nous vous recommandons chaudement. *Ils se dévorent, même si vous n'êtes pas un grand lecteur de roman !*

03 – Rendre votre travail : La méthode NFS

Toute l'année durant, vous aurez à rendre des travaux. La méthode de rendu peut varier. Dans un premier temps, pour cette période de voyage en hyper-espace, la méthode de rendu sera la « **Méthode NFS** ».

Les fichiers que vous manipulez sur vos postes sont en effet en vérité situé sur notre serveur, sur lequel nous avons un contrôle total. Par conséquent, nous pourrons ramasser vos travaux.

Nous vous prions de bien vouloir considérer la chose suivante : nous ne regarderons **jamais** le *contenu* de vos fichiers en dehors des fichiers que nous devons ramasser et qui seront clairement annoncés à chaque fois, **mais** nous vous encourageons tout de même à conserver votre espace scolaire exempt d'éléments extra-scolaire.

Sachez que cet espace est limité en taille, donc nous nous réservons le droit de supprimer des fichiers d'une taille supérieure à 500Mo. Vous serez bien sur prévenu préalablement afin de pouvoir sauvegarder votre fichier. Si vous avez au contraire une large collection de petits fichiers occupant un espace considérable, nous vous demanderons de procéder vous-même à leur suppression afin de vous éviter une perte.

04 – Propreté de votre rendu

Dans le dossier de rendu, seul votre travail a sa place. Cela signifie que la présence de tout fichier excédentaire vous portera préjudice car cela **arrêtera** votre évaluation.

Il peut y avoir **plusieurs évaluations par jour**, et donc tout ne sera pas perdu : vous aurez l'opportunité de réparer votre erreur et d'attendre la prochaine, néanmoins prenez garde car si celle que vous avez raté était la **dernière**, il n'y en aura pas d'autre.

Les fichiers typiquement interdits sont les fichiers terminant par ~ ou commençant par **#** qui peuvent être laissé par votre éditeur de texte favoris, **emacs**. N'hésitez pas à les supprimer avec la commande **rm**. Si vous ne savez pas vous servir de **rm**, ouvrez son manuel en tapant la commande « **man rm** » dans votre terminal.

Sont également interdits, mais vous ne serez concerné que plus tard, les fichiers terminant en **.o**, **.so**, **.a**, et plus généralement les **artefacts et produits de compilation** (c'est à dire, vos propres programmes construits, seules les sources et ressources nous intéressent)

05 – La ligne de commande

Ouvrez un terminal. Le terminal est comme une fenêtre d'explorateur de fichier, sauf que celui-ci fonctionne en texte. Par défaut, votre terminal vous place dans votre « maison », dans votre dossier personnel.

Vous pouvez observer la position de votre dossier personnel dans le système de fichier en tapant la commande **pwd** *(print working directory)*. Voici ce qu'il arrive lorsque je tape moi-même **pwd**. Pour information, le symbole **« $> »** représente le « **prompt** », il s'agit d'un simple indicateur marquant le début de l'endroit où l'utilisateur tape sa commande.

```
$> pwd
/home/daemonlab
$>
```

Tapez vous-même la commande **pwd** afin d'observer ce qu'il se passe.

Vous pouvez lister le contenu du dossier courant en utilisant la commande **ls** *(list)*. Il est possible de passer des arguments à une commande. Pour cela il suffit de séparer les arguments par des espaces. Ci-dessous, un exemple avec un **ls** simple suivi d'un exemple avec un **ls** augmenté de l'option **-l** qui permet de lister plus d'informations sur les fichiers.

```
$> ls
Pictures code.c file.
$> ls -l
drwxr-xr-x 2 daemonlab daemonlab 4096 march 12 2021 Pictures
-rw-rw-rw- 1 daemonlab daemonlab  128 march 12 2021 code.c
```

Vous pouvez entrer dans un dossier à l'aide de la commande **cd** *(change directory)*. Celle-ci prend en paramètre un chemin de fichier. Vous pouvez donc faire ceci :

```
$> pwd
/home/daemonlab
$> cd Pictures
$> pwd
/home/daemonlab/Pictures
$> cd ..
$> pwd
/home/daemonlab
```

Pour indiquer un chemin montant et revenir dans le dossier parent, le symbole est « **..** ».

Mais que sont les commandes **ls**, **pwd** et **cd** ? Les commandes ls et **pwd** sont des programmes ! Taper leur nom les invoque. La commande cd elle, fait partie de la ligne de commande elle même. Qu'est ce que cela signifie ? Que vous pouvez appeler n'importe quel logiciel depuis la ligne de commande. Tapez **firefox** pour voir !

Vous allez maintenant ouvrir votre éditeur de texte favoris : **emacs.** Son rôle est de vous permettre d'écrire du texte dans un fichier. Inutile de chercher à utiliser la *souris*, **emacs** ne réagit qu'au **clavier** !

Pour l'ouvrir, c'est très simple : comme n'importe quel programme, il suffit de taper son nom ! Lancez-le… puis quittez le. Pour quitter, appuyez sur **CONTROL**, que vous garderez enfoncé le temps de taper sur **X** *puis* sur **C**.

Si vous souhaitez écrire dans un fichier, il vous suffira de passer le nom du fichier que vous souhaitez manipuler en paramètre. Si il n'existe pas, il sera crée à la sauvegarde. Tentez par exemple de créer le fichier **identity** et écrivez votre **login** dedans suivi d'un saut de ligne. Pour sauvegardez, appuyez sur **CONTROL**, que vous garderez enfoncé le temps de taper sur **X** puis sur **S**. Quittez ensuite **emacs**.

La commande **cat** permet d'ouvrir les fichiers passés en paramètre et de les afficher dans le terminal. Vous pouvez vérifier ainsi le contenu de votre fichier. L'option **-e** permet de faire apparaître plus clairement les caractères spéciaux comme le sauts de ligne. Ainsi, en plus du saut de ligne lui-même apparaîtra avec l'option **-e** le symbole **$** symbolisant le saut de ligne.

```
$> cat -e identity
daemonlab$
$>
```

Vous allez placer ce fichier **identity** dans un dossier situé dans **~/hyperespace/jprep/** (~ représentant votre dossier personnel, cela veut dire que vous devez créer un dossier **hyperespace** dedans, puis dans ce dossier **hyperespace** créer un dossier **jprep** dans lequel vous devrez mettre le fichier **identity**)

Comment créer un dossier ? Utilisez la commande **mkdir** *(make directory)*.
Comment déplacer le fichier **identity** ? Utilisez la commande **mv** *(move)*.
Comment apprendre à utiliser les commandes **mkdir** et **mv** ? Utilisez la commande man : « **man mkdir** », « **man mv** » vous montreront les manuels des commandes. Vous voulez en savoir plus sur les manuels ? « **man man** ». Ceci n'est pas une blague.

La présence dans le bon dossier du fichier identité et la justesse de son contenu est votre **premier exercice**.

06 – Liste des fichiers cachés

Écrivez dans le fichier **~/hyperespace/jprep/lsall.sh** une commande permettant de lister tous les fichiers *y compris les cachés (un fichier est caché si son nom commence par un point)* du dossier courant en mode liste (Option **-l** vue précédemment). La commande devra s'arranger pour que les dossiers soient suivis du symbole « **/** » et que la taille des éléments affichés ne soit pas en octet mais affiché d'une manière plus lisible, exploitant les notations kilo, mega, giga.

Comment lister les fichiers du dossier courant ? Vous devriez le savoir.

Comment s'arranger pour que la commande ai le comportement attendu ? En découvrant ses options à l'aide de son manuel.

Pour tester la commande écrite dans le fichier, vous pouvez utiliser la commande **sh** qui employé avec un fichier procède à l'exécution des commandes qu'il contient.

```
$> pwd
/home/daemonlab/hyperespace/jprep/
$> sh lsall.sh
drwxrwxr-x 4 daemonlab daemonlab 4,0K april 26 2020 ./
drwxr-xr-x 5 daemonlab daemonlab  20K sept. 24 2020 ../
drwxr-xr-x 2 daemonlab daemonlab 4,0K march 12 2021 Pictures/
-rw-rw-rw- 1 daemonlab daemonlab  128 march 12 2021 code.c
```

Vous allez écrire un second fichier dans le **même** dossier, cette fois appelé **lstilde.sh** et qui s'occupera de lister tous les fichiers terminant par un tilde, le fameux ~. Pour cela, vous utiliserez le symbole *****. Ce symbole a pour particularité de pouvoir remplacer une série de 0 caractère ou plus de n'importe quelle nature et de s'étendre en fonction des fichiers et dossiers du dossier courant. Par exemple, pour déplacer tous les fichier **.c** dans le dossier **Pictures**, nous pourrions faire :

```
$> mv *.c Pictures/
$> ls
Pictures
$> ls Pictures
code.c
```

Écrivez donc la commande qui permet d'afficher tout ce qui termine par **tilde** dans **ldtilde.sh**.

07 – La configuration de bash

Qu'est ce que **bash** ? C'est le nom du programme que vous utilisez depuis le début pour lancer des commandes ! Le **terminal** est une boite de texte, le **shell** est l'interprète de commande. Ici, le **shell** s'appelle **bash**. Il en existe d'autres, mais c'est celui-ci que vous retrouverez le plus souvent.

Ce shell utilise le fichier **.bashrc** comme fichier de configuration. Qu'est ce qu'un fichier de configuration ? C'est un fichier qui contient des commandes qui sont exécutées lorsque **bash** se lance : lorsque vous ouvrez votre **terminal** par exemple.

Il existe une commande que vous trouverez en abondance dans ce fichier de configuration, c'est la commande **alias**. Cette commande n'est pas un logiciel comme **ls**, mais une fonctionnalité de **bash** lui-même, comme l'est la commande **cd**. La commande **alias** permet de définir une nouvelle commande qui sera en fait… une autre commande maquillée.

```
$> alias bonjour="echo -e bonjour $USER\nCa va ?"
$> bonjour
bonjour daemonlab
Ca va ?
$>
```

Ici, nous avons fabriqué une commande **bonjour**. Quand on tape **bonjour**, cela exécute « **echo -e bonjour $USER\nCa va ?** ».

La commande **echo** sert à afficher le texte passé en paramètre suivi d'un saut de ligne. L'option **-e** permet d'interpréter certaines suites de caractères, **\n** signifiant **« saut de ligne »**.

« **$USER** » est une *variable d'environnement*, c'est à dire une valeur situé dans le shell lui-même. En l'occurrence, cette variable d'environnement contient votre login. Si vous souhaitez voir la liste des variables d'environnement, il vous suffit de taper la commande **env**.

Vous allez ajouter à votre configuration **bash** une commande **tidy** qui va supprimer les fichiers terminant par ~ dans le dossier courant et **dans tous les sous-dossiers.** Pour pouvoir écrire cet alias, vous aurez besoin d'utiliser la commande **find**.

Vous vous rappelez comment faire pour vous renseigner sur une commande ?

08 – Chaînage d'opérations

Vous trouverez sur **l'intranet** un fichier **database.csv** contenant des informations. Téléchargez le et placez le dans le dossier de rendu **~/hyperespace/jprep** avant de vous lancer dans l'écriture d'une nouvelle commande. Celle-ci sera contenue dans le fichier **get_nickname**.

La commande que vous devez faire doit ouvrir le fichier **database.csv** et afficher le **pseudonyme** (troisième colonne) de la personne dont le nom est passée en paramètre.

Comment utiliser un paramètre dans un script shell (ce que vous êtes en train de faire) ? En utilisant la variable **$1**, vous récupérerez le premier argument lancé au script. (**$2** le deuxième, etc.)

Votre fichier commencera cette fois par le **shebang** suivi du programme qui exécutera le fichier : */usr/bin/sh*.

https://fr.wikipedia.org/wiki/Shebang

La préparation de ce fichier avec le **shebang** vous permettra de l'exécuter avec **sh** sans avoir à taper « **sh get_nickname** » comme vous l'avez fait précédemment. Le **shebang** permettant de dire « **exécute ce fichier avec ce programme** » au système d'exploitation.

Néanmoins, comment faire ? Taper **« get_nickname »** ne permet pas de lancer le fichier : en effet, le shell pense qu'il s'agit d'une commande et il n'y a pas de commande qui s'appelle **get_nickname**. Vous devez taper à la place **« ./get_nickname »**, en étant dans le dossier (ou alors, vous devez taper le chemin complet vers le fichier, **« . »** signifiant **« ici »**) pour signifier au shell que ce n'est pas une commande mais un fichier que vous souhaitez exécuter.

Et ca ne marche toujours pas ! Pourquoi ? Parce que votre fichier n'est pas exécutable. Vous n'avez pas le droit de l'exécuter. Comment faire pour pouvoir l'exécuter ? Donnez vous le droit de le faire en utilisant la commande **chmod**. Vous savez ou trouver le manuel.

Une fois que le fichier est exécutable et que vous savez comment l'appeler… il ne se passe rien, étant donné que votre fichier ne contient pas encore de commande…

Le shell vous permet de chaîner les commandes : à l'aide de l'opérateur **« | » (pipe)**, vous pouvez lancer une commande, qui écrira des choses, mais au lieu de les écrire sur le terminal, le fera sur l'entrée d'un autre programme qui de ce fait traitera les informations. Exemple avec la commande **rev** qui inverse les octets des lignes d'un flux de données.

```
$> echo -e "Salut\nCa va?" | rev
tulaS
?av aC
$>
```

Pour réussir cet exercice, vous aurez besoin :

- D'afficher le contenu du ficher, ce que vous devriez déjà savoir faire
- De filtrer en fonction du contenu d'une ligne (commande **grep**)
- De couper la ligne pour sélectionner une sous partie (commande **cut**)