Survive a night out . . . UNIX version !

Le principe

Les rôles

Plus on est de fous, plus on rit ! Il y a donc deux rôles :

- **Celui qui exécute les ordres** : c'est celui derrière l'ordinateur. Lui seul peut vous aider à rentrer chez vous sain et sauf.
- **Celui qui les donne** : c'est vous ! Assumez votre esprit de leader et tentez de rentrer chez vous le plus vite possible avant de vous endormir.

Vous devez coopérer pour comprendre ce que l'un et l'autre doivent faire !

Les modules

À travers 8 épreuves, vous aurez l'occasion de découvrir des commandes UNIX différentes :

- *réveil difficile*
- *les poches pleines*
- *où est ma carte étudiante ?*
- *le mélange parfait*
- *l'adresse de la soirée*
- *le digicode*
- *le concierge*
- *c'est quelle clé, déjà ?*

Chaque module a une certaine difficulté, mais soyez-en sûrs : plus vous jouerez, plus vous y arriverez !

L'ordre dans lequel vous résolvez les modules n'a pas vraiment d'importance, mais c'est toujours plus fun de suivre l'histoire dans l'ordre !

Comment gagner ?

Reconstruisez votre aventure et **trouvez les indices avant la fin du compte à rebours** pour vivre une bonne soirée !

Attention cependant aux contrecoups de votre semaine chargée : **plus vous mettez de temps à résoudre le niveau, plus le risque d'endormissement est fort**, et **si le nombre d'erreurs est élevé, alors surchauffe du cerveau assurée**.

Chaque fois que vous vous tromperez, la matière grise de votre cerveau se détériore, et une erreur sera comptabilisée. Chaque niveau a un nombre d'erreurs autorisées différent. Ce nombre vous sera donné au début de chaque activité de la soirée. Si vous arrivez au nombre d'erreurs maximum autorisé, l'épuisement sera annoncé, même s'il vous reste encore du temps !

Les commandes

Il y a de nombreuses commandes à connaître, et c'est parfois compliqué de s'y retrouver. C'est pourquoi il y a un manuel à la fin du document !

Les commandes de base à savoir

- Lancer un script `bash` : `./<nom_script.sh>` *Les scripts bash ont généralement une extension en* `.sh`
- Lister tous les fichiers : `ls`
- Se déplacer dans un répertoire :
  - `cd <nom_du_répertoire>` pour entrer dans le répertoire
  - `cd ..` pour remonter au répertoire d'au-dessus
- Afficher le contenu d'un fichier : `cat <nom_du_fichier>`

Comment savoir combien de temps il me reste ?

1. Naviguez jusque dans le répertoire principal `jeu`
2. Affichez le contenu du fichier `time` avec la commande `cat`

Bon courage, et bonne soirée !

Réveil difficile

Compétences utilisées

- Allumage de l'ordinateur sur un environnement Linux (avec une clé USB)
- Connexion au WI-FI (eduroam)
- Téléchargement des commandes et des fichiers nécessaires
- Lancement du jeu

Bienvenue dans cette épopée !

Le contexte

Nous sommes jeudi, il est 18 heures, et vous venez de vous réveiller de votre sieste bien méritée après une semaine chargée. Toujours un peu dans vos rêves, vous recevez un message... mais impossible de l'ouvrir ! Vous avez un trou de mémoire : vous ne vous souvenez plus du mot de passe de votre téléphone... ni de votre ordinateur. (C'était le même, ce n'est pas très malin...)

Heureusement, vous avez récemment appris à allumer votre ordinateur sur un environnement Linux grâce à une clé USB. Il est temps de mettre vos compétences à l'épreuve et de nous montrer de quoi vous êtes capable.

Connexion au réseau

Toujours pas de réseau ? Connectez-vous au Wi-Fi via `eduroam`.

Installation des commandes nécessaires

```
nom_de_la_commande:  command not found
```

Comment régler cela ?

1. Passez en mode super-utilisateur :

- `sudo -s`

2. Mettez à jour la liste des paquets :
   - `apt update`
3. Installez la commande manquante :
   - `apt install nom_de_la_commande`

Installation du jeu depuis GitHub

1. Rendez-vous dans le dossier souhaité :
   - `cd /chemin/vers/le/dossier`
2. Clonez le dépôt GitHub : `git clone https://github.com/lilound/G_2_NADLER_Lilou.git`
3. il faut ensuite dézipper le fichier jeu.zip. Pour ce faire : tapez dans le terminal `tar -xvzf fic.tar.gz -C dossier_jeu`
4. Entrez dans le dossier du jeu :
   - `cd dossier_jeu`
5. changez les permissions de tout le dossier `dossier_jeu`. Il faut pouvoir exécuter chaque fichier. La commande chmod le fait. Tapez la commande `find . -type f -name "*.sh" -exec chmod +x {} \;`

Vous voilà prêt.e pour le lancement du jeu !

Problème de place – les poches pleines

Commandes utilisées

- `cd`
- `ls`
- `cat`
- `rm`

Commencez par vous diriger vers le dossier `poches` à l'aide de la commande `cd`.

Vous n'avez pas besoin de tout ça !

Non, vous ne partez pas en randonnée pour une semaine : vous sortez juste pour une soirée. Voyons voir ce que vous avez dans les poches...

Affichez leur contenu avec la commande `ls`.

Il semble que vous ayez des objets en double... Pourquoi faire ? Supprimez tous les doublons en gardant chaque objet unique avec le plus petit numéro. Utilisez la commande `rm`.

Exemple de noms de fichiers : `1_cle.txt, 2_cle.txt, 3_verre.txt` → vous garderez `1_cle.txt, 3_verre.txt`

Lancement du module

Pour lancer le module, en lançant le script: `start_poches.sh`

Une fois le script lancé, différents fichiers apparaîtront dans le répertoire.

Vous ne savez pas comment lancer un script ? → **Référez-vous à la section Commandes de base.**

Vérification de la réponse

Si vous êtes sûr · e de vous, lancez le script suivant :

- `./verification_poches.sh`

Résultat :

- Si la réponse est correcte → message `"Bravo !"`
- Sinon → message `"Dommage !"`

Où est ma carte étudiante ??

Commandes utilisées

- `find`
- `pwd`

Le principe

Vous avez perdu votre carte étudiante... elle est quelque part dans vos poches, mais où ! À vous de la retrouver, et prouver que vous êtes bien étudiant · e.

Lancement du module

Pour démarrer le module : lancez le script `start_carte.sh`

Étapes à suivre

1. Retrouvez le fichier contenant votre carte étudiante. Indice : son nom est très explicite...

Vérification de la réponse

Pour vérifier votre réponse : lancez le script `verification_carte.sh`. On vous demandera le chemin relatif au dossier poches de carte pour accéder à votre carte étudiante.

- Si la réponse est correcte → `"Bravo !"`
- Sinon → `"Dommage !"`

L'adresse de la soirée

Commandes utilisées

- `mkdir`
- `cp`
- `mv`
- `cat`
- `vi`

Le principe

Vous êtes motivé.e, il est temps d'y aller, et ce soir vous êtes d'humeur à faire la fête. Saurez-vous retrouver les indices sur l'adresse exacte de la soirée? Pour ce faire, il faut suivre une série d'étapes dans le terminal, et surtout : savoir écrire dans un fichier avec `vi`.

Lancement du module

Pour démarrer le module, lancez le script `start_soiree.sh`

Une fois le script lancé, un fichier nommé `indice.txt` apparaît dans le répertoire. A vous de le vérifier pour retrouver le lieu.

Étapes à suivre

1. Créer un répertoire nommé `soiree`
2. Dans ce même répertoire, copier le fichier `adresse.txt` dans ce répertoire et renommez le en `lieu.txt`
3. Modifier le contenu du fichier `infos.txt` avec `vi`
4. Écrire les informations exactes de l'after dans le fichier sous la forme **bâtiment, heure, salle** Exemple : `K-Fêt INSA, 20h30, salle du bas`
5. Sauvegarder et quitter
6. Vérifier le contenu en affichant `infos.txt`

Vérification de la réponse

Quand vous êtes prêt · e, lancez le script `verification_soiree.sh`

Résultat

- Si la réponse est correcte → `"Bravo !"`
- Sinon → `"Dommage !"`

Le mélange parfait

Commandes utilisées

- `ps aux`
- `kill`
- `htop`
- `grep`
- `head`

Le principe

Enfin arrivé.e à la soirée ! Vous avez commandé des boissons avec vos amis, mais étourdi.e comme vous êtes, vous avez oublié laquelle vous aviez prise! Heureusement, votre choix est stocké dans votre ordinateur, à un endroit dont vous ne vous seriez peut-être pas douté... dans vos processus en cours. Chaque processus porte un nom de boisson (`mojito.sh`, `pina_colada.sh`,...). Votre mission : identifier le bon cocktail à servir (ou à éliminer) selon les critères donnés dans le fichier `recette.txt`.

Lancement du module

Pour démarrer le module :

- `./start_melange.sh`

Une fois lancé, plusieurs scripts s'exécutent en arrière-plan, chacun représentant un cocktail.

Étapes à suivre

1. Afficher tous les processus en cours
2. Filtrer les processus liés aux cocktails en ne gardant que ceux avec les ingrédients présents dans la boisson
3. Identifier le bon cocktail selon les critères du fichier `recette.txt` :
4. Vérifier maintenant le CPU du processus retenu.
   - Si le cocktail est trop fort (CPU $>$ 30%), il faut l'"adoucir" :
     - `kill -STOP <PID>`
   - Si le cocktail est trop léger (CPU $<$ 5%), il faut le "booster" :
     - `kill -CONT <PID>`

Vérification de la réponse

Quand vous pensez avoir servi le bon cocktail : lancez `verification_melange.sh`

Réinitialisation du module

Résultat

- Si la réponse est correcte → `"Bravo !"`
- Sinon, le cocktail est raté (nom incorrect ou mauvaise recette), il faut le "jeter" :
  - `kill <PID>`

Le digicode de l'entrée

Commandes utilisées

- `ls`
- `grep`
- `cat`
- `| (pipe)`
- `wc`
- `cut`

Le principe

Il est temps de rentrer, vous ne trouvez pas ? Vous voila arrivé.e devant votre immeuble... mais la porte est verrouillée par un digicode qui a été changé la semaine dernièe! Quoi? Vous ne vous en souvenez plus? C'est pas grave, on va y remédier... Le code est dissimulé dans plusieurs fichiers répartis dans le répertoire `porte`. À vous de retrouver les bons fragments, les assembler, et taper le code final !

Lancement du module

Pour démarrer le module lancez le programme `start_digicode_1.sh`

Une fois lancé, plusieurs fichiers apparaissent dans le dossier `porte`, contenant des fragments de code, des leurres, et des indices.

Étapes à suivre

1. Afficher les fichiers présents
2. Rechercher les fichiers contenant le mot-clé `code` et qui ne contiennent qu'un chiffre (pas d'espace, pas de lettres, juste un chiffre).
3. Assembler les chiffres dans l'ordre indiqué par le fichier `ordre.txt` :

Astuce : Certains fichiers peuvent contenir des pièges ou des fragments inutiles. Ne vous fiez pas aux apparences !

Vérification de la réponse

Quand vous avez reconstitué le digicode, lancez le script `verification_digicode_1.sh` et tapez le digicode dans la console

Résultat

- Si le digicode est correct → `"Bravo !"`
- Sinon → `"Dommage !"`

Le concierge de l'immeuble

Commandes utilisées

- `apt-get install`
- `wget`
- `ping`
- `> (redirection)`
- `ip link`
- `file`
- `du`

Le principe

Vous avez franchi le digicode... mais ce n'était qu'un sas. Après avoir mis autant de temps à trouver le code, le concierge de l'immeuble s'inquiète : habitez-vous vraiment ici? Pour lui prouver, vous devez l'épater. Pour continuer, vous devez prouver que vous savez interagir avec le réseau, télécharger des fichiers, et analyser leur contenu. Un fichier vous attend quelque part sur Internet. À vous de le récupérer et d'en extraire la bonne information.

Lancement du module

Pour démarrer le programme `start_digicode_2.sh`

Une fois lancé, un fichier `mission.txt` apparaît avec des instructions précises sur l'action à réaliser.

Étapes à suivre

1. Lire le fichier `mission.txt` :
2. Si une commande est manquante, installez-la.
3. Indice : Si la mission demande de tester une connexion : il est recommandé d'utiliser les paramètres -c 3 dans la commande

Vérification de la réponse

Une fois la mission accomplie, lancez le script `verification_digicode_2.sh` et écrivez la commande effectuée.

Résultat

- Si la réponse est correcte → `"Bravo !"`
- Sinon → `"Dommage !"`

C'est quelle clé, déjà...?

Commandes utilisées

- `find`
- `pwd`
- `echo $PATH`
- `export`

Le principe

Vous voila enfin arrivé.e devant votre porte !! Vous avez plusieurs clés... mais une seule ouvre la bonne porte. Laquelle? Chaque clé est un fichier placé dans un dossier différent, et seule celle dont le chemin est correctement ajouté à la variable `PATH` permet de valider le module.

Lancement du module

Pour démarrer le module : lancez le programme `start_cles.sh`.

Une fois lancé, plusieurs dossiers contenant des fichiers `cle.sh` sont créés. Un fichier `indice.txt` vous aide à identifier la bonne clé.

Étapes à suivre

1. Lire le fichier `indice.txt` pour identifier la bonne clé
2. Rechercher tous les fichiers `cle.sh`
3. Se rendre dans le dossier contenant la bonne clé
4. Afficher son chemin absolu
5. Ajouter ce chemin à la variable `PATH`
6. Vérifier que la variable a bien été modifiée

Attention : Ne remplacez pas entièrement la variable `PATH`, sinon les anciens chemins seront perdus et la réponse ne sera pas comptabilisée comme correcte.

Vérification de la réponse

Quand vous êtes prêt · e, lancez le script `verification_cles.sh`

Résultat

- Si la réponse est correcte → `"Bravo !"`
- Sinon → `"Dommage !"`

Explications des commandes

Les commandes

Les commandes de bases à savoir

`./script.sh`

Lancer un script : `./script.sh` Les scripts bash ont généralement une extension en `.sh`.

`ls`

Exemple : `ls` (liste les fichiers d'un répertoire)

`pwd`

Exemple : `pwd` (affiche le chemin du répertoire courant)

`cat`

Pour afficher le contenu d'un fichier : `cat fichier.txt`

La commande `ls` et ses options

La commande `ls` permet d'afficher les éléments du répertoire courant. Voici quelques options importantes :

- `ls -l` : liste les fichiers avec informations détaillées
- `ls -a` : liste les fichiers, y compris les fichiers cachés
- `ls -lh` : liste les fichiers avec taille lisible (Ko, Mo)
- `ls -lt` : liste les fichiers triés par date de modification
- `ls -ltr` : liste les fichiers triés par date de modification (ordre inverse)
- `ls -lS` : liste les fichiers triés par taille
- `ls -lSr` : liste les fichiers triés par taille (ordre inverse)

Ces options peuvent bien sûr être combinées : `ls -al`

Création d'un répertoire

`mkdir`

Pour créer un répertoire : `mkdir nom_du_répertoire` Exemple : `mkdir mon_dossier`

Copie d'un fichier

`cp`

Pour copier un fichier : `cp fichier_source fichier_destination` Exemple : `cp fichier.txt mon_dossier/fichier.txt`

Renommer un fichier

`mv`

Pour renommer un fichier : `mv ancien_nom nouveau_nom` Exemple : `mv fichier.txt nouveau_fichier.txt`

Supprimer un fichier

rm

Pour supprimer un fichier : `rm fichier.txt`

Installation d'un paquet via apt

apt

La commande `apt` (Advanced Package Tool) est utilisée pour gérer les paquets dans les distributions basées sur Debian (comme Ubuntu). Elle permet d'installer, de mettre à jour, de supprimer et de rechercher des paquets logiciels.

- Mettre à jour les paquets : `sudo apt update && sudo apt upgrade`
- Installer un paquet : `sudo apt install nom_du_paquet`
- Supprimer un paquet : `sudo apt remove nom_du_paquet`
- Rechercher un paquet : `apt search nom_du_paquet`

Les redirections

>

Les redirections avec le chevron `>` permettent d'envoyer la sortie d'une commande vers un fichier.

Exemple :

```
ls > output.txt
```

La sortie est enregistrée dans le fichier `output.txt`.

|

La redirection avec le symbole `|` (pipe) permet d'envoyer la sortie d'une commande comme entrée d'une autre.

Exemple :

```
ls -l | grep ".txt"
```

```
Commande 1   |   Commande 2
ls -l        |   grep ".txt"
```

Autre exemple :

```
cat fichier.txt | grep "texte_recherche"
```

Éditeur de texte : `vi`

Mode Normal (déplacement)

Ce mode permet de se déplacer dans le texte. - Pour passer en mode normal : touche `Échap`. - Déplacements : `k` (haut), `j` (bas), `h` (gauche), `l` (droite). - Début de ligne : `0` - Fin de ligne : `$`

Supprimer une ligne

En mode normal, placer le curseur sur la ligne et taper `dd`.

Supprimer un caractère

En mode normal, placer le curseur sur le caractère et taper `x`.

Sauvegarder et quitter

- Sauvegarder et quitter : `:wq` - Quitter sans sauvegarder : `:q!`
- Sauvegarder sans quitter : `:w`

La commande `grep`

La commande `grep` (Global Regular Expression Print) permet de rechercher des chaînes de caractères dans un fichier ou dans la sortie d'une commande. Elle est souvent utilisée avec des pipes pour filtrer les résultats.

Exemple :

```
grep "texte" fichier.txt
```

La commande `ps aux`

La commande `ps aux` liste les informations complètes des processus en cours d'exécution sur le système, avec des détails comme l'identifiant du processus (PID), l'utilisateur ainsi que la commande lancée.

Exemple :

```
ps aux | grep "nom_du_processus"
```

La commande `kill`

La commande `kill` permet d'envoyer un signal à un processus pour le terminer ou le relancer. Par défaut, elle envoie le signal TERM (15) qui demande l'arrêt du processus.

Options utiles :

- `kill PID` : termine le processus avec le PID spécifié
- `kill -9 PID` : envoie le signal KILL pour forcer l'arrêt immédiat
- `kill -l` : liste tous les signaux disponibles

Exemple :

```
kill 1234
```

La commande `htop`

`htop` est un outil interactif et visuel pour la gestion des processus et l'utilisation des ressources du système. Il permet de voir les processus en cours, leur consommation CPU/mémoire, et de les trier ou tuer facilement.

Exemple :

```
htop
```

La commande `find`

La commande `find` permet de rechercher des fichiers et des répertoires dans l'arborescence selon des critères spécifiques.

Syntaxe de base :

```
find [chemin] [critères] [actions]
```

Critères courants :

- `-name` : recherche par nom
- `-type` : recherche par type (f pour fichier, d pour répertoire)

- `-size` : recherche par taille
- `-mtime` : recherche par date de modification

Actions :

- `-print` : affiche les fichiers trouvés
- `-exec` : exécute une commande sur les fichiers trouvés

Exemples :

```
find / -name "fichier.txt"
find . -type f -size +10M
find /home/user -mtime -7
find . -name "*.log" -exec rm {} \;
```

La commande `ping`

La commande `ping` est utilisée pour tester la connectivité réseau entre l'hôte local et une adresse IP ou un nom de domaine. Elle envoie des paquets ICMP et mesure le temps aller-retour.

Syntaxe :

```
ping [options] destination
```

Options courantes :

- `ping -c 4 google.com` : spécifie le nombre de paquets à envoyer
- `ping -i 2 google.com` : définit l'intervalle entre paquets
- `ping -t 64 google.com` : définit le TTL (nombre de sauts maximum)
- `ping -s 128 google.com` : définit la taille des paquets
- `ping -W 2 google.com` : délai d'attente pour une réponse
- `ping -q google.com` : mode silencieux (affiche uniquement le résumé)
- `ping -f google.com` : envoie des paquets aussi vite que possible (mode flood)
- `ping -n google.com` : affiche les adresses IP au lieu des noms d'hôtes
- `ping -4 google.com` : force l'utilisation d'IPv4
- `ping -6 google.com` : force l'utilisation d'IPv6

—

La commande `pwd`

La commande `pwd` (Print Working Directory) affiche le chemin absolu du répertoire courant.

Exemple :

```
pwd
# Sortie : /home/utilisateur/projet
```

—

La commande `export`

La commande `export` permet de définir des variables d'environnement, accessibles dans le shell et ses sous-processus.

Syntaxe :

```
export VARIABLE=valeur
```

Utilisations principales :

1. Personnalisation de l'environnement utilisateur
2. Communication entre processus
3. Contrôle des configurations des applications
4. Gestion des ressources système

`wget` Télécharge un fichier depuis Internet.

- wget https://example.com/fichier_secret.txt
- Télécharge le fichier `fichier_secret.txt` depuis l'URL indiquée.
- Le fichier est enregistré dans le répertoire courant.

`file` Analyse le type d'un fichier.

- file fichier_secret.txt
- Affiche le type MIME ou descriptif du fichier (ex. `ASCII text`).
- Utile pour identifier la nature d'un fichier inconnu.

`du` Mesure la taille d'un fichier ou dossier.

- du -h fichier_secret.txt
- Affiche la taille du fichier dans un format lisible (ex. `4.0K`).
- L'option `-h` signifie *human-readable*.

`ip link` Liste les interfaces réseau disponibles.

- ip link
- Affiche toutes les interfaces réseau (ex. `lo`, `eth0`, `wlan0`).
- Chaque interface est décrite avec son état, son nom et son adresse MAC.

Exemples de variables d'environnement courantes :

- `PATH` : répertoires contenant les exécutables
- `HOME` : répertoire personnel de l'utilisateur
- `USER` : nom de l'utilisateur
- `SHELL` : shell utilisé par l'utilisateur
- `LANG` : langue et paramètres régionaux