

TP 3 & 4 de familiarisation avec Unix

Droits – Archives – Meta-caractères – Filtres – Processus

Vincent Berry - vberry@lirmm.fr

Préambule

Important : avant de commencer cette feuille de TP, vous devez faire de bout en bout la section 3 ("intégration des notions vues sur le SGF") de la feuille de TP 2.

Rappel : si vous découvrez dans ce module l'environnement Unix, vous aurez tout intérêt à éviter de passer du temps sur les paragraphes dont le titre est précédé d'un symbole **@** (pour expert). Ces paragraphes pourront être examinés en dehors de la présente séance. **Si vous connaissez déjà le système Unix, vous devez passer par toutes les questions, y compris celles avec ce symbole.**

Table des matières

1 Les commandes minimum sur les processus (10 mn)	1
1.1 Processus et commandes	1
1.2 Interrompre un processus lancé par le terminal	1
1.3 Gestion des processus @	2
2 Groupes, droits des fichiers : exploration plus avancée du SGF (≈ 1h30)	2
2.1 La logique des droits (15mn)	2
2.2 Travail en groupes (1h plus ou moins 5mn)	2
2.3 Contenu public et contenu privé (15 mn)	5
3 Redirections et filtres (≈ 30 mn)	5
3.1 Utilisation courantes (25 mn)	5
3.2 Redirections et filtres avancés @ (10 mn)	6
4 Archivage (≈ 30 mn)	7
4.1 Premières manipulations (5 mn)	7
4.2 Application grandeur réelle (25 mn)	7
5 @ Identification de l'utilisateur et de son environnement (5mn)	8

1 Les commandes minimum sur les processus (10 mn)

1.1 Processus et commandes

A l'exécution de tout programme, le Système d'exploitation crée un **processus**, sorte d'entité qui enregistre les informations sur l'exécution de ce processus (sous quelle identité s'exécute-t-il, à quelle instruction en est-il, etc).

Quand vous lancez une commande dans un terminal, vous provoquez la création d'un processus. Vérifiez en ouvrant une nouvelle fenêtre Terminal, ou un nouvel onglet dans la fenêtre Terminal actuelle. Demandez la liste des processus

(commande `ps`) avec l'option permettant de voir les filiations entre processus). Indiquez les bons paramètres pour voir les processus que vous avez lancés non seulement dans le shell en cours (par défaut) mais aussi ceux lancés dans d'autres shells (option `-u`).

Essayez la commande `ps tree <votrelogin>` qui vous permet de voir explicitement la filiation entre vos processus.

1.2 Interrompre un processus lancé par le terminal

Certaines commandes tapées dans un Terminal ne se terminent pas tout de suite, ceci peut être embêtant. Dans un terminal, lancez la commande `xemacs`. Revenez dans le Terminal et essayez maintenant la commande `ls` sur le répertoire en cours.

Normalement, tapez cette dernière commande ne provoque rien dans votre terminal. La raison en est que la commande précédente (`xemacs`) n'a pas encore terminé.

En tapant `CTRL - D` dans le terminal, vous récupérez la possibilité de taper de nouvelles commandes : le système a forcé le processus qui était en cours d'exécution à se terminer (brutalement). Du coup, si vous aviez commencé à taper des modifications dans un document important, vous perdez ces modifications qui ne seront pas répercutées dans le fichier sur disque.

`CTRL - Z` permet d'interrompre un processus de façon temporaire. Faites un essai en relançant d'abord `emacs`.

Pour qu'il continue de s'exécuter en arrière plan, vous devez taper `bg` (pour *background*, c-a-d *arrière-plan*). Dans ce cas, vous gardez la main dans le terminal (enfin, c'est une façon imagée de dire que vous pouvez encore y taper des commandes).

Vous devriez aussi aimer la possibilité que le terminal vous laisse de lancer d'emblée un processus en arrière-plan : pour cela il suffit d'ajouter le caractère `&` à la fin de la commande saisie dans le terminal. Faites un essai de lancement de `xemacs` de cette façon et vérifiez que vous conservez la possibilité d'exécuter des commandes dans le terminal.

1.3 Gestion des processus @

La commande `top` vous donne un aperçu des processus en cours d'exécution (les plus gourmands en ressources sont listés en premier). Accessoirement, la commande vous fournit aussi l'état de la mémoire centrale.

La commande `ps` vous permet de connaître les processus lancés depuis votre terminal et encore en cours d'exécution. Les options `-x` et `-a` vous permettent de voir plus de processus. L'option `--forest` est aussi très intéressante. Faites des essais et reportez-vous au `man` pour bien maîtriser ces options.

Enfin, vous pouvez expérimenter la commande `kill` au nom très explicite. Bien-sûr vous ne pouvez tuer que des processus tournant sous votre identité.

2 Groupes, droits des fichiers : exploration plus avancée du SGF (≈ 1h30)

2.1 La logique des droits (15mn)

Placez-vous dans une fenêtre Terminal pour réaliser les manipulations suivantes :

- Positionnez-vous dans le répertoire `Unix` qui est juste sous la racine de votre *votre* espace personnel.
- Créez un répertoire `Droits` et descendez dans ce répertoire.
- Créez un fichier texte `faire-cafe.txt` avec l'éditeur (x)emacs. Dans ce fichier vous indiquerez (succinctement) les opérations à réaliser pour préparer un bon café (une instruction par ligne, maximum 4 lignes). Quittez ensuite emacs après avoir sauvegardé votre prose.
- Affichez l'ensemble des informations disponibles sur les fichiers de votre répertoire courant.
- Quels sont l'utilisateur et le groupe propriétaire du fichier `faire-cafe.txt` ?
Quels sont leurs droits sur ce fichier ?
- Essayez d'exécuter ce fichier (taper `./faire-cafe.txt` dans le terminal).
Que signifie le message que le terminal vous renvoie ?
Pourquoi un tel message ? Quels droits seraient nécessaires pour que le fichier soit exécutable ?
- Changez les droits du fichier pour qu'il soit exécutable par l'utilisateur propriétaire (`chmod`)
- Faites une copie du fichier `./faire-cafe.txt` sous le nom `recette.txt` (commande `cp`). Quels sont les droits du nouveau fichier ? Pouvez-vous en déduire la logique du Système d'Exploitation lors de la création du nouveau fichier ?

- Essayez à nouveau d'exécuter le fichier `./faire-cafe.txt`. Pouvez-vous expliquer la raison précise des messages qui résultent de cette tentative ?
- Pour le fichier `faire-cafe.txt`, en une seule commande, donnez tous les droits au groupe, aucun à l'utilisateur et aux autres (`chmod`).
- Essayez maintenant de visualiser le contenu de ce fichier (commande `cat` ou par `emacs`). Que constatez-vous ? Pourquoi ?
- Essayez de détruire ce fichier (`rm`). Vous devez constater que vous arrivez à réaliser cette opération (après éventuellement avoir confirmé l'opération au shell par `y` suivi de `<Entreé>`). Par quel miracle cela est-il possible avec les droits actuels sur ce fichier ?

2.2 Travail en groupes (1h plus ou moins 5mn)

Préparons le terrain (20 - 25mn)

- Créez un répertoire `PUBLIC` juste sous la racine de votre arborescence personnelle.
- Changez le groupe propriétaire de ce répertoire pour le groupe nommé `alternatif`¹ (commande `chgrp`, voir le man pour le détail des arguments, ou tout simplement l'invoquer avec l'option `--help`).
- Comment vérifier que le changement a bien eu lieu ?
- *Note : suivant comment a été installé le Linux des salles TP (et comment sont gérés les fichiers par NFS, l'opération précédente peut ne pas aboutir. Ce n'est pas grave, passez à la suite.*
- Donnez-vous tous les droits sur ce répertoire `PUBLIC`, ainsi qu'au groupe, mais uniquement les droits de traverser et lire le contenu du répertoire aux autres personnes (c-a-d `others`). Vérifiez.
- Mais au fait, à quel(s) groupe(s) appartenez-vous ? (`id`)
- Indiquez au shell que vous voulez maintenant qu'il vous identifie comme une personne appartenant au groupe `alternatif` (`newgrp alternatif`, mot de passe éventuel)
- Vérifiez qu'il a tenu compte de votre changement de groupe (`id` éventuellement avec l'option `-g`)
- Allez dans votre répertoire `PUBLIC` et créez-y un sous-répertoire `PROJET`.
- Donnez tous les droits sur ce répertoire à l'utilisateur et au groupe propriétaires (c-a-d `alternatif`), mais aucun droit aux autres personnes. Vérifiez.
- Allez dans le sous-répertoire `PROJET`
- Créez dans ce sous-répertoire un fichier texte du nom de `projet.txt`
- Après avoir réfléchi un minimum et avoir fait un aller-retour avec les diapos du cours, tapez dans le fichier `projet.txt` quelques lignes de texte dans un format concis indiquant la procédure à adopter à votre avis pour que l'ouverture de toute fenêtre de Terminal affiche un message de bienvenue personnalisé².
- Revenez dans le terminal et vérifiez à quel utilisateur et à quel groupe appartient le fichier que vous venez de créer (commande `ls` avec la bonne option).
- Sortez du shell dans lequel la commande `newgrp` vous a placé à votre insu (`exit`). Méditez la phrase précédente et si vous ne la comprenez pas bien, n'hésitez pas à mettre au défi votre enseignant de vous fournir une explication précise, y compris en termes de processus.
- Vérifiez que vous avez repris votre groupe de départ (`id` éventuellement avec l'option `-g`).

Collaborons (10 mn)

Supposons que vous participiez au projet mis en place par votre voisin. Vous allez avoir besoin d'accéder aux fichiers et répertoires du projet, afin d'avancer le travail lié à ce projet. Nous allons d'abord vérifier quelles manipulations vous pouvez effectuer sur ces fichiers et répertoires.

Tournez-vous vers l'étudiant(e) de l'ordinateur d'à côté et échangez vos noms de login. Déduisez-en le chemin menant à son répertoire d'accueil. Vérifiez.	
Essayez de consulter le contenu du fichier <code>projet.txt</code> qu'il a créé quelque part sous son répertoire <code>PUBLIC</code> . Pourquoi n'est-ce pas possible ?	

¹pour les besoins du TP, tous les étudiants font partie de ce groupe, mais pensez que pour un projet à réaliser, vous pouvez demander la création d'un groupe Unix limité aux étudiants de votre projet.

²indiquer les idées que vous avez, même si vous ne les avez pas vérifiées pour l'instant et n'en êtes pas très sûr.

Identifiez-vous maintenant comme appartenant au groupe <code>alternatif</code> (<code>newgrp</code> , vérifiez avec <code>id</code>)	
Pouvez-vous maintenant voir le contenu de son fichier <code>projet.txt</code> ? Pourquoi ?	
Pouvez-vous créer une copie de ce fichier (nommée par exemple <code>copie.txt</code>) dans son répertoire <code>PROJET</code> (<code>cp</code>) ?	
invoquez la commande <code>ls</code> avec la bonne option pour connaître les utilisateur et groupes propriétaires des fichiers du répertoire <code>PROJET</code> . Qui est propriétaire du fichier <code>copie.txt</code> ? A votre avis, quel utilisateur voit diminuer son quota disque en raison de ce fichier <code>copie.txt</code> situé dans l'arborescence de votre voisin ?	
Pouvez-vous détruire le fichier <code>projet.txt</code> (<code>rm</code>) ? Comment cela est-il possible ?	
Si vous êtes parvenus à cet exploit, renommez le fichier <code>copie.txt</code> en <code>projet.txt</code> (commande <code>mv</code> , et non pas la commande <code>cp</code>)	

Notions de protocole et de synchronisation (15 - 20 mn)

Maintenant que nous avons vérifié que vous avez effectivement la possibilité de changer le contenu du projet, vous allez avancer l'état dans lequel se trouve le projet. Pour cela, rien de plus compliqué : ajoutez quelques idées et commentaires dans la feuille de route du projet, c-a-d le fichier `projet.txt` de votre voisin.

Mais au fait, que pourrait-il se passer si un autre voisin de votre voisin essaye de faire la même opération que vous sur le fichier de votre voisin (c-a-d celui que vous êtes en train d'éditer, vous suivez ☺) ?	
Est-ce que vous auriez évité ces soucis si vous aviez adopté le protocole suivant ? <ol style="list-style-type: none"> 1. faire une copie du fichier <code>projet.txt</code> dans un fichier <code>monlogin.txt</code> 2. éditer cette copie du fichier original 3. remplacer le fichier <code>projet.txt</code> par la copie modifiée (<code>monlogin.txt</code>) ? 	
Décrivez un scénario qui met en défaut le protocole défini ci-dessus	

Pourtant il existe une façon de procéder plus élaborée que celle proposée ci-dessus, mais néanmoins simple, qui vous permet d'éviter ces soucis d'accès concurrent à un même fichier (indice : utiliser la commande <code>mv</code>). Groupez-vous à trois postes pour établir un protocole évitant toute perte de travail effectué par un collaborateur dans le fichier <code>projet.txt</code> . Vérifiez que le protocole fonctionne, et essayez de convaincre votre enseignant que ce protocole est robuste ³ .	
---	--

Bravo, non seulement, vous venez de concevoir un protocole, mais vous avez vu un problème fondamental en informatique : la synchronisation des accès à une ressource partagée.

Un peu de ménage (10 mn)

- Après avoir édité le fichier `projet.txt` (suivant un protocole défini ci-dessus), vérifiez dans le terminal que vos changements ont été pris en compte. Expérimentez à cette occasion la commande `less` à la place de la banale commande `cat`. Comment localiser rapidement avec cette commande les lignes que vous avez ajoutées ?
- Après votre intervention dans le projet situé chez votre voisin, est-il possible de remettre le contenu de son répertoire `PROJET` exactement dans l'état dans lequel il était avant que vous y fassiez les manipulations décrites ci-dessus ? Pourquoi ?
- Revenez dans votre propre répertoire `PROJET` et regardez si vos voisins y ont créé des fichiers. Notamment, regardez si votre fichier `projet.txt` existe encore et demandez à voir son contenu pour voir si vos voisins y ont fait des contributions intéressantes.
- Après avoir vérifié que vos voisins ont fini les manipulations de votre projet décrites dans ce TP (ou celles inspirées par leur soif de connaissance), vérifiez si le fichier `projet.txt` vous appartient encore. Si ce n'est pas le cas, faites les manipulations nécessaires (copie, suppression, changement de nom) pour que ce fichier vous appartienne à nouveau et pour supprimer toute trace du passage de vos collaborateurs dans votre répertoire.

2.3 Contenu public et contenu privé (15 mn)

- Dans votre répertoire d'accueil (`~`), créez un répertoire de nom `PRIVE`. Positionnez les droits de ce répertoire pour que seul l'utilisateur propriétaire aie tous les droits dessus, et que les autres entités (groupe et autres) n'aient que les droits de traverser ce répertoire (rappel : droit `x`). Vérifiez.
- Convenez avec votre voisin de gauche d'un nom de répertoire pas trop évident⁴ pour stocker votre nouveau projet ultra secret et créez dans le répertoire `PRIVE` un sous-répertoire au nom choisi. Donnez les droits `777` à ce sous-répertoire. A quoi correspondent ces droits ?
- Demandez à votre voisin de gauche (oui, le même que précédemment) d'accéder à votre répertoire. Peut-il y arriver ? Et en persévérant ?
- Peut-il créer un fichier ou un répertoire dans le répertoire correspondant à votre projet secret ? Pourquoi ?
- Bien, vous désirez maintenant être sûr que ce projet n'est bien accessible que des personnes connaissant l'existence du projet et le nom du répertoire en question. Faites rapidement le vide dans votre Terminal (`clear`) pour qu'aucun indice visuel ne subsiste sur votre écran. Ensuite, communiquez votre nom de login (mais pas le nom de votre projet secret !) à votre voisin *de droite* (oui, celui qui est un concurrent notoire de votre projet secret) et mettez le au défi d'essayer d'accéder depuis son ordinateur à votre nouveau projet secret.
- S'il n'y parvient pas, est-ce lié au fait qu'il est sur un ordinateur différent ?
- Est-ce lié aux droits du sous-répertoire de `PRIVE` qui correspond à votre projet secret ?
- Est-ce lié aux droits du répertoire `PRIVE` ?

⁴Évitez l'évident "secret"

3 Redirections et filtres (≈ 30 mn)

3.1 Utilisation courantes (25 mn)

La commande `ypcat passwd` et la commande `ypcat hosts` indiquent la liste des utilisateurs, resp. des machines, connus du système d'identification. Essayez.

Pour connaître les informations associées à votre identité, nous allons sauvegarder les informations renvoyées par la commande précédente dans un fichier <code>utilisateurs.txt</code> . Quelle ligne de commande faut-il entrer dans le terminal pour réaliser cette opération ?	
Vérifiez le contenu du fichier créé dans le terminal (commande <code>less</code>).	
Comment utiliser la commande de filtre <code>egrep</code> pour localiser dans ce fichier la ligne qui vous concerne ?	
Essayez de décrypter les quelques informations qui sont indiquées pour vous sur cette ligne.	
Quel enchaînement de commandes aurait permis de visualiser cette ligne, sans passer par un fichier intermédiaire ? (utilisation d'un tube). Essayez.	
Comment envoyer la ligne résultant de cette commande dans un fichier <code>/UNIX/mesInfos.txt</code> ?	
Comment ajouter en une commande la date d'aujourd'hui et l'heure à la fin du fichier <code>mesInfos.txt</code> (commande <code>date</code>) sans recourir à un copier/coller ou à un éditeur ?	
En analysant plus finement le résultat de la commande <code>ypcat passwd</code> , comment savoir maintenant combien d'étudiants d'IG portent le même <i>prénom</i> que vous ? (commande <code>egrep</code>)	
L'étudiant Paul Riquet veut savoir combien de personnes portent le même <i>nom</i> que lui. Il utilise pour cela l'expression régulière suivante "Riquet", mais obtient en réponse la ligne le concernant plus un certain nombre de lignes non désirées comme Jean.Riquette. Comment cela se fait-il ?	
Pour parvenir à un résultat correct, il utilise maintenant le motif suivant : <code>^[A-Z]\.Riquet</code> . Pouvez-vous indiquer pourquoi il utilise un symbole <code>\</code> devant le caractère "." ?	
Quand il teste sa commande avec le motif indiqué ci-dessus, il n'obtient pas une seule ligne de résultat, même pas celle correspondant à son nom de login. Pouvez-vous indiquer quelle erreur il a faite ?	

Après avoir corrigé son erreur, il obtient bien les lignes souhaitées, mais toujours accompagnées de plusieurs lignes non correctes, comme celle du dénommé Riquette . Indiquez comment il doit compléter son motif actuel pour n'avoir que les lignes des utilisateurs ayant exactement le même nom que lui.	
Maintenant que nous avons un bon motif pour cette personne, pouvez-vous le tester pour votre nom. Quel résultat ?	
Est-il possible d'obtenir le même résultat correct avec un motif plus court ? si oui, lequel ?	

3.2 Redirections et filtres avancés @ (10 mn)

Nous allons maintenant examiner quelques commandes permettant de retrouver des fichiers.

Consultez le manuel de la commande `find`. A quoi cette commande sert-elle ?

Faites un essai en vous positionnant dans votre répertoire d'accueil et en lui demandant de localiser le fichier `projet.txt` (utilisez l'option `-name`)

Sans changer de répertoire, demandez à cette commande de localiser dans l'ensemble de l'arborescence du SGF (c-a-d depuis /) le programme `emacs`.

Comment filtrer les lignes qui ne nous intéressent pas (correspondant à des répertoires dans lesquels vous n'avez pas le droit d'aller) en se servant de la commande `grep` ?

Comme alternative, il est possible de rediriger la sortie des erreurs vers le faux fichier `/dev/null` en ajoutant `2>/dev/null!` à la fin de la commande `find`. Faites un essai pour trouver la syntaxe exacte et vérifiez que le résultat de la commande est maintenant plus lisible.

Question subsidiaire : pouvez-vous trouver l'option de `find` qui permet de lister tous les fichiers modifiés depuis hier ?

4 Archivage (≈ 30 mn)

4.1 Premières manipulations (5 mn)

- Dans un Terminal, placez-vous dans votre répertoire d'accueil
- Créez une **archive unix.tar** comprenant tous les fichiers de votre répertoire UNIX (commande `tar` avec l'option `c`)
- Vérifiez que le fichier a bien été créé dans votre répertoire d'accueil (`ls` avec le bon argument) et qu'il contient bien les bons répertoires et fichiers (commande `tar` avec l'option `t`)
- Essayez de détruire le répertoire UNIX. Que faut-il faire pour y parvenir ?
- Restaurez l'arborescence du répertoire UNIX en désarchivant le fichier `unix.tar` : `tar -xvf <nomarchive>`
- Explorez le répertoire restauré ainsi que ses sous-répertoires pour vérifier que tout est bien en place.
- Faites une copie du fichier `unix.tar` sur une clef usb et essayez de restaurer cette archive plus tard chez vous.

4.2 Application grandeur réelle (25 mn)

La suite de votre contrat pour le projet **Musiversal**, implique d'intégrer de nouveaux modules dans l'application actuelle

- Si ce n'est pas déjà le cas, déplacez le projet **Musiversal** dans votre répertoire `Unix` puis déplacez-vous dans le répertoire `Unix`
- Créez une archive `musiversal-v1.tar` pour sauvegarder le contenu de ce projet et vérifiez ensuite qu'elle contient bien tout ce qu'elle doit (`tar` option `t`)
- Téléchargez maintenant le fichier `neoModule.tar.gz` et placez-le dans le répertoire `Unix`.
- D'après l'extension de ce fichier, sous quel format sont stockées les informations qu'il contient ?
- Décompressez le contenu de ce fichier (`gzip` avec la bonne option). Vérifiez le résultat de cette opération. Combien de fichiers ont été créés ? Le fichier `neoModule.tar.gz` existe-t-il toujours ?

- Demandez à consulter le contenu de l'archive `.tar` que vous avez obtenu (commande `tar` avec la bonne option)
- D'après le chemin indiqué pour les fichiers contenus dans cette archive, quel est le répertoire dans lequel vous devriez placer cette archive avant d'essayer de la désassembler ?
- Après avoir éventuellement déplacé l'archive, désassemblez les fichiers qu'elle contient de façon à ce qu'ils se situent au bon niveau en comparaison des autres modules
- Vérifiez en demandant le contenu des répertoires concernés.
- Donnez les droits d'exécution au fichier programme contenu dans ce nouveau module que vous venez de récupérer.
- Ajoutez le nom du module dans la liste des modules de l'application, placez une copie de son programme dans le répertoire `bin` et lancez une exécution de l'application pour voir si tout fonctionne bien.
- De même, ajoutez la documentation de ce nouveau module à la fin de la documentation actuelle, en utilisant une redirection (`>>`).

Note : si vous rencontrez un problème lors d'une des étapes précédentes, pensez à faire le ménage et à repartir de l'application qui fonctionnait, stockée dans l'archive `musiversal-v1.tar`

Il est question de déployer cette application chez le client, mais l'entreprise ne souhaite livrer que les exécutables et les fichiers de configuration nécessaires.

- Faites la liste de ce qu'il est nécessaire d'envoyer chez le client, puis dans le répertoire `UNIX` créez un répertoire de même nom que votre login et envoyez-y une copie des fichiers que vous avez repérés. Recréez exactement la sous-arborescence qui est nécessaire aux exécutables de l'application, c-a-d en omettant le répertoire `Musiversal` lui-même (mais pas tout son contenu) et les répertoires de développement.
- Testez l'exécutable dans ce nouvel endroit. Si cela ne fonctionne pas demandez de l'aide à votre enseignant après avoir examiné le contenu des scripts et essayé de repérer vous même pourquoi votre installation locale ne fonctionne pas.
- Quand cette installation fonctionne, créez une archive `exploitation.tar` qui contient votre installation de l'exécutable (c-a-d le répertoire de même nom que votre login et son contenu). Vous pouvez procéder en une seule exécution de la commande `tar` (en indiquant moult arguments) ou bien en complétant progressivement l'archive (voir le `man` de la commande `tar`), mais faites en sorte qu'au désassemblage de votre archive, l'ensemble aille bien dans un répertoire à votre nom.
- Vérifiez que cela va être le cas en demandant la liste et le chemin des fichiers et répertoires de l'archive (`tar` option `t`)
- Comprimez maintenant l'archive pour passer pour un vrai pro avant de la transmettre (commande `gzip`).
- Pour vérifier que vous avez fait du bon travail, le patron vous demande de lui envoyer votre archive avant d'envoyer au client. Prudent, vous préférez d'abord tester avec un collègue pour éviter de prendre trop de risques avec le big boss : rédigez un fichier `README.txt` qui contient la commande pour désarchiver votre archive et la commande à lancer ensuite pour exécuter l'application⁵, et envoyez un mél à votre voisin de gauche ou de droite avec ce fichier d'explication en attaché, ainsi bien-sûr que l'archive compressée.
- Demandez à votre collègue d'opérer une décompression et un désassemblage de l'archive dans un répertoire `tmp` qu'il aura créé dans son répertoire `Unix`, mais séparément de son répertoire `Musiversal` (afin que l'application que vous délivrez soit vraiment testée en solo, comme sur une machine ne contenant pas l'ensemble du projet `Musiversal`).
- Après avoir effectué cette vérification Ne quittez pas la salle de TP avant d'avoir montré à votre enseignant (votre chef de service) qu'effectivement votre archive de l'exécutable est opérationnelle chez un collègue⁶.

5 @ Identification de l'utilisateur et de son environnement (5mn)

- Qui suis-je? (`whoami`, `id`)
- Suis-je le seul connecté sur ma machine? (`who`)
- D'ailleurs quel est le nom de ma machine? (`hostname`)
- Quelle est l'adresse intranet de ma machine? (`yptest machine hosts`)
- Combien de place mémoire me reste-t-il sur disque? (`du`, `quota`)
- Quel est l'état de la mémoire centrale? (`top`) (`q` pour arrêter l'exécution de la commande)
- Combien de place mémoire est-ce que j'utilise sur disque? (`du`)
- Ai-je un quota d'espace disque que je peux utiliser, et à combien est-il rempli? (`quota`)

⁵Soyez sûrs de bien exécuter la manoeuvre, car il vous faudra la répéter pour rendre un certain nombre de projets durant votre parcours IG

⁶autrement dit, en environnement hostile ☹