R1.04 – Cours 5

Droits d'accès, liens symboliques, scripts, compléments

Département Informatique

IUT2, UGA

2023/2024

Plan du cours

IUT2 Université Grenoble Alpes
Département INFO

Plan du cours

IUT2 Université Grenoble Alpes
Département INFO

Attributs d'un fichier (métadonnées)

- Nom
- Type (fichiers, dossier, ...)
- Taille
- Date de dernière modification
- ...
- **Propriétaire (UID)**
- **Groupe (GID)**
- **Permissions / droits d'accès**
- ...

IUT2 Université Grenoble Alpes Département INFO

Visualiser UID,GID et les permissions

Commande : `ls -l`

```
toto@transit:~$ ls -l /users/info/pub/1a/R1.04/tp-fichiers/
total 80
-rw-r--r-- 1 bonnaudl info 14859 Sep  9  2013 Avare.txt
drwxr-xr-x 2 bonnaudl info  4096 Oct 11 19:42 dossier
-rw-r--r-- 1 bonnaudl info   413 Sep 30 11:29 Essai.class
-rw-r--r-- 1 bonnaudl info   131 Sep 30 11:27 Essai.java
-rwxr-xr-x 1 bonnaudl info    17 Oct 25 11:25 essai-script
-rw-r--r-- 1 bonnaudl info   430 Oct  8 13:25 Prg.class
-rw-r--r-- 1 bonnaudl info   153 Oct  3 13:13 Prg.java
-rw-r--r-- 1 bonnaudl info   282 Oct  3 13:11 Proverbes.txt
```

IUT2 Université Grenoble Alpes Département INFO

Interpréter les permissions

- On distingue 3 catégories d'utilisateurs
  - `u` (**user**)
    désigne la personne qui a créé le fichier ou le répertoire (son propriétaire)
  - `g` (**group**)
    désigne les membres du groupe d'utilisateurs auquel le fichier appartient (infobut1, infobut2, profs, ...)
  - `o` (**others**)
    désigne tous les autres.
- Il existe 3 sortes de permissions
  - `r` (read) : permission de **lecture**
  - `w` (write) : permission d'**écriture**
  - `x` (execute) : permission d'**exécution** ou de **traversée**
- Il existe $3 \times 3 = 9$ permissions pour chaque entrée

IUT2 Université Grenoble Alpes Département INFO

Exemples de permissions

```
toto@transit:~$ ls -l /users/info/pub/1a/atelier-Linux/
total 40
-rw-r--r-- 1 bonnaudl info 14859 Aug 30 Avare.txt
drwx------ 2 bonnaudl info  4096 Aug 30 dossier-protégé/
-rw------- 1 bonnaudl info    20 Aug 30 fichier-protégé.txt
-rw-r--r-- 1 bonnaudl info   282 Aug 30 proverbes.txt
-rw-r--r-- 1 bonnaudl info 11913 Oct  6 Tux.png

toto@transit:~$ ls -l /usr/bin/cp
-rwxr-xr-x 1 root root 151168 Sep 24  2020 /usr/bin/cp
```

IUT2 Université Grenoble Alpes
Département INFO

Permissions et chemin d'accès

- Exemple de chemin d'accès
  `/users/info/pub/1a/atelier-Linux/Avare.txt`
- Permissions nécessaires pour donner l'accès en lecture
  - Permission `x` sur `/users`
  - Permission `x` sur `info`
  - ...
  - Permission `x` sur `atelier-Linux`
  - Permission `r` sur `Avare.txt`
- Si **une seule** de ces permissions manque, l'accès est **refusé**

IUT2 Université Grenoble Alpes Département INFO

Sémantique des permissions

Permission	Lecture (`r`)	Écriture (`w`)	Exécution (`x`)
Fichier	consulter le contenu (afficher, copier, ...)	modifier le contenu	exécuter (le fichier doit être un programme ou un script)
Répertoire	consulter la liste des entrées qu'il contient (`ls`, ...)	modifier la liste des entrées qu'il contient (créer une entrée, renommer, supprimer)	traverser le répertoire (utilisation dans un chemin d'accès à une entrée)

IUT2 Université Grenoble Alpes Département INFO

Changer les permissions (1/2)

`chmod [-R] [QUI]+|-PERM ENTREE...`

- `QUI`
  - à qui s'applique le changement de permissions
  - chaîne fabriquée avec les caractères `u g o` (ou chaîne vide)
- `PERM`
  - quelles permissions on veut ajouter ou enlever
  - chaîne fabriquée avec les caractères `r w x X`

Exemples pour un fichier

- `chmod g+r f1`
  autorise les membres du groupe de `f1` à lire `f1`
- `chmod o-w f1`
  interdit aux autres d'écrire dans `f1`
- `chmod go-rwx f1`
  interdit aux non propriétaires tous les accès à `f1`
- `chmod +x f1`
  rend `f1` exécutable

IUT2 Université Grenoble Alpes Département INFO

Changer les permissions (2/2)

Exemples pour un répertoire

- `chmod g+rx rep1`
  autorise les membres du groupe de rep1 à lister le contenu de `rep1` et à traverser `rep1`, mais pas sa descendance.
- `chmod -R g+rx rep1`
  idem, mais y compris la descendance.
  Attention : tous les fichiers deviennent exécutables !
- `chmod -R g+rX rep1`
  idem, mais seuls les fichiers déjà exécutables par le propriétaire sont rendus exécutables pour le groupe.

IUT2 Université Grenoble Alpes Département INFO

Plan du cours

Rappel : types d'entrées

- Fichiers
- Répertoires
- "Fichiers" représentant les périphériques (*devices*)
- **Liens symboliques**
- Autres
  - Tubes nommés
  - Sockets

Exemples de liens symboliques

- Exemples vus avec `ls -l`
- Type d'entrée repéré avec la lettre `l`, couleur bleu clair
- Répertoire `/usr/local/bin/`

```
lrwxrwxrwx 1 root staff 34 Sep  6 idea-2022 -> ../idea-IC-212.5080.55/bin/idea.sh
lrwxrwxrwx 1 root staff 34 Sep  6 idea-2023 -> ../idea-IC-222.3739.54/bin/idea.sh
lrwxrwxrwx 1 root staff 34 Sep  6 idea -> idea-2023
```

- Répertoire `/usr/bin/`

```
lrwxrwxrwx 1 root root 37 Jun 25 x-terminal-emulator -> /etc/alternatives/x-terminal-emulator*
```

- Répertoire `/etc/alternatives/`

```
lrwxrwxrwx 1 root root 31 Jun 28 x-terminal-emulator -> /usr/bin/gnome-terminal*
```

- Répertoire `/users/info/pub/bin/`

```
lrwxrwxrwx 1 bonnaudl info 50 Jun 29 trie-DNS -> /users/info/pub/2a/reseaux/tp_exploration/trie-DNS*
```

IUT2 Université Grenoble Alpes Département INFO

Utilisation, propriétés

- Principe
  accéder à un fichier ou répertoire depuis un autre chemin que le chemin d'origine
- Utilité
  permet d'accéder à une entrée,
  sans avoir besoin d'en faire une copie
- Peut pointer vers un fichier, un répertoire, ou un autre lien symbolique
- Peut servir d'aide mémoire ou de "raccourci"
- Peut être absolu ou relatif
- Un lien affiché en rouge est un lien qui pointe vers une entrée qui n'existe pas
- Un lien a toujours des permissions `rwxrwxrwx`
  - Elles ne sont pas utilisées
  - Ce sont les permissions de la cible qui sont utilisées

IUT2 Université Grenoble Alpes
Département INFO

Commande de création

- Logiciel : `ln` (*link*)
- Commande : `ln -s` (s comme "symbolique")
- Lien avec le même nom que l'entrée d'origine

  `ln -s ENTREE_EXISTANTE REPERTOIRE_OÙ_CRÉER_LE_LIEN`
- Lien avec un autre nom que l'entrée d'origine

  `ln -s ENTREE_EXISTANTE NOM_DU_LIEN`
- Exemples

```
cd /usr/local/bin/
ln -s ../idea-IC-212.5080.55/bin/idea.sh .
```

⟶ le lien créé s'appelle `idea.sh`

```
ln -s ../idea-IC-212.5080.55/bin/idea.sh idea
```

⟶ le lien créé s'appelle `idea`

IUT2 Université Grenoble Alpes Département INFO

Plan du cours

Scripts *shell*

- Principe : exécution séquentielle automatique de commandes
- Exemple : fichier texte nommé `ménage-java`

```
#!/bin/bash
# Ce script supprime tous les fichiers
# non indispensables
#   (ceci est un commentaire)

# Supression des fichiers compilés
rm *.class
# Supression des fichiers de backup
rm *~
```

IUT2 Université Grenoble Alpes Département INFO

Création d'un script

1. Créer un fichier texte avec un éditeur de texte (gedit, ...)
2. Écrire le texte du script
3. Rendre le fichier texte exécutable

```
chmod +x NOM_DU_SCRIPT
```

⟶ Exécuter le script depuis **le répertoire où il est stocké**

```
./NOM_DU_SCRIPT
```

⟶ Exécuter le script depuis **n'importe quel répertoire**

- Mettre le script dans un répertoire où le *shell* le trouvera automatiquement

```
NOM_DU_SCRIPT
```

IUT2 Université Grenoble Alpes
Département INFO

Répertoires standards de stockage des scripts

Scripts pour un utilisateur particulier

- Répertoire `~/.local/bin/`
- Contenu dans le *homedir* de l'utilisateur
- L'utilisateur peut y mettre ce qu'il veut

Scripts pour tous les utilisateurs du système

- Répertoire `/usr/local/bin/`
- Seul le super-utilisateur (`root`) y a accès

IUT2 Université Grenoble Alpes
Département INFO

Variables du *shell*

- 2 types de variables

Variables d'environnement

- Existent pour tout processus (cf. `/proc/PID/environ`)
- Transmises d'un processus père à ses fils
- Nom généralement en majuscules
- Modifient le comportement de certains logiciels

Variables locales à un *shell*

- Définies uniquement pour un processus *shell* particulier
- Ses processus fils n'en héritent pas
- Nom généralement en minuscules

- Transformation variable locale $\longrightarrow$ variable d'environnement

```
export NOM_DE_VARIABLE
```

- Définition directe d'une variable d'environnement

```
export NOM_DE_VARIABLE=VALEUR
```

IUT2 Université Grenoble Alpes Département INFO

Variables d'environnement couramment rencontrées

- Commande pour lister les variables : `env`
- Quelques variables standard

```
SHELL=/bin/bash
TERM=xterm-256color
USER=toto
PATH=/usr/local/bin:/usr/bin:/bin:
     /usr/games:/users/info/pub/bin
PWD=/users/info/etu-1a/toto/R1.04
PS1=\[\033]0;[\u@\h:\w]\007\]\u@\h:\w\$
HOME=/users/info/etu-1a/toto
```

- Préférences personnelles

```
PAGER=less
EDITOR=gedit
BROWSER=firefox
```

IUT2 Université Grenoble Alpes Département INFO

Rôle de la variable `PATH`

- Liste de répertoires où le *shell* cherche des programmes ou scripts
- Pour un simple utilisateur
  répertoires `bin`
- Pour le super-utilisateur (`root`)
  répertoires `bin` et `sbin`

IUT2 Université Grenoble Alpes
Département INFO

Utilisation de variables dans un script

- Affectation
  - Exemple

    ```
    fichier="/tmp/toto"
    ```
  - Attention aux espaces !

- Utilisation
  - Nom de la variable précédé par `$`
  - Guillemets pour le cas où la variable contient des caractères spéciaux
  - Exemple 1 :

    ```
    echo -n "Nom de votre fichier : "
    echo "$fichier"
    ```
  - Exemple 2 : le *shell* remplace la variable par sa valeur

    ```
    echo "Nom du fichier: $fichier"
    ```
  - Cela permet de faire facilement des **concaténations**

IUT2 Université Grenoble Alpes Département INFO

Utilisation de variables dans un script

- Déclaration : pas de déclaration !
  - Une simple affectation suffit
  - Attention à ne pas se tromper de nom !
  - Exemple

```
i=0
echo "$j"  # Pas de message d'erreur.
           # Rien n'est affiché.
```

- Type : pas de type !
  - Exemple

```
i="toto"
i=9
```

  - Interprétation selon le contexte
  - Pas de compilation, erreurs détectées lors de l'exécution

IUT2 Université Grenoble Alpes Département INFO

Passage de paramètres à un script

- Utilisation de variables particulières prédéfinies
- Nom de ces variables : `$1`, `$2`, ..., `$9`
- Paramètres (positionnels) passés au script
  depuis la ligne de commande du script vers les variables `$1`, `$2`, ...
- Exemple de passage de paramètre à un script

```
$ ./menage-java toto
```

- Exemple de récupération de paramètre dans un script

```
#!/bin/bash
rm "$1.class" "$1.java~"
```

- À l'exécution, le *shell* remplace `$1` par `toto`

⟶ `rm toto.class toto.java~`

IUT2 Université Grenoble Alpes Département INFO

Autres variables particulières prédéfinies

- `$0` : nom du script. Ex :

```
echo "Bonjour, je suis le script $0"
```

- `$#` : nombre de paramètres passés au script.
- `$@` : liste des paramètres passés au script.
  Équivalent à `"$1" "$2" ...`
- `$?` : code (numérique) de retour passé au système par la dernière commande appelée. En général 0 si pas d'erreur, nombre $\neq$ 0 en cas d'erreur. Ex :

```
cp "$1" "$2"
echo "code de retour de la commande cp: $?"
```

- `$$` : PID du *shell* exécutant le script

IUT2 Université Grenoble Alpes Département INFO

Structures de contrôle

- Instruction conditionnelle :
  `si ... alors ... sinon ...`
- Boucles `for`
- Boucles `while`
- ...
- Langage puissant, mais parfois déroutant
- Tout ceci sera vu en 2ème année (parcours B)...

IUT2 Université Grenoble Alpes
Département INFO

Caractères spéciaux du *shell*

- Caractères à éviter dans les noms de fichiers
- Les espaces
- Les jokers (*wildcards*) : `* ? [ ]`
- Les 3 types de guillemets
  - *double quote* `(")` utiles pour utiliser des espaces et la plupart des autres caractères spéciaux. Les variables sont encore substituées.
  - *single quote* `(')` utiles pour tous les caractères spéciaux. Les variables ne sont plus substituées.
  - *back quote* `(`)` permet de lancer une commande et de récupérer son résultat. Ex :

```
date_courante=`date -I`
ou date_courante=$(date -I)    (version plus moderne)
```

- Le *backslash* `(\)` pour rendre normal un caractère spécial

```
$ echo \"
"
```

- Autres caractères spéciaux : `! ; ( ) | & $ < > #`

Plan du cours

Initialisation du *shell*

- Scripts communs à tous les utilisateurs
  - `/etc/profile`
  - `/etc/bash.bashrc`
- Scripts spécifiques à un utilisateur
  - `~/.profile`
  - `~/.bashrc`
  - `~/.bash_logout`
- Scripts par défaut utilisés lors de la création d'un compte
  répertoire `/etc/skel/` (*skeleton*)

IUT2 Université Grenoble Alpes Département INFO

Réglages avancés du *shell*

- Ce sont des réglages de type on/off
  qui modifient le comportement du *shell*
- Commande interne : `shopt` (*shell option*)
- Liste des réglages

```
$ shopt
autocd          off
cdable_vars     off
cdspell         off
cmdhist         on
[...]
```

- Nombre de réglages

```
$ shopt | wc -l
57
```

Réglages avancés du *shell*

- Signification des réglages : `man bash`

```
The list of shopt options is:

autocd
  If set, a command name that is the name of a directory
  is executed  as  if it  were  the argument to the cd command.
  This option is only used by interactive shells.

cdable_vars
  If  set,  an  argument to the cd builtin command
  that is not a directory is assumed to be the name
  of a variable whose value is the directory to change to.
[...]
```

- Activation d'un réglage

  `shopt -s NOM_RÉGLAGE` (*set*)

- Désactivation d'un réglage

  `shopt -u NOM_RÉGLAGE` (*unset*)

IUT2 Université Grenoble Alpes Département INFO

Exemple de réglage intéressant : `globstar`

- Activation :
  ```
  shopt -s globstar
  ```
- Utilisation :
  nouveau *wildcard* `**`
- Usage : aller chercher des fichiers dans toute une arborescence quelque soit le niveau de profondeur
- Exemple :
  ```
  rm  ~/**/*.class  ~/**/*~
  ```

IUT2 Université Grenoble Alpes Département INFO

Plan du cours

Résumé

- Je fais attention aux droits d'accès de mes fichiers
- Je sais interpréter et créer des liens symboliques
- Les scripts *shell* sont très utiles pour de petits programmes, mais attention à la syntaxe !