

Formation DevOps

Penser à bien valider sa présence sur : <http://quest.ajc-formation.fr:8000> (matin et après-midi).

Code wifi : 7894561230

Algo et Objet

Prof. : ASSOUS Steeve

Code : 3404

Conception d'algo :

ANALYSE -> CONCEPTION -> PROGRAMMATION -> TEST

En POO :

Objet contient 3 concepts fondamentaux :

- **polymorphisme**
- **encapsulation**
- **héritage**

Objets de bases :

- données en entrée et résultats intermédiaires
- règle opératoire
- ici on parle du nom commun objet (différent de POO)

Un objet contient :

- un **identificateur** (nom)
- un **type** (entier, numérique, caractère ...)
- une **valeur**

Types simples

Ce sont les **caractères**, **textes** (ou **chaînes de caractères**), **logiques**, **bouléens**, **indicateurs** et les **nombres**.

Types composés

Ce sont les **fiichiers** et les **tableaux**.

Actions de base

Elles sont :

- affectation : attribuer une valeur à un objet
- affectation particulière : initialisation, incrémentation/décrémentation

Ex :

```
// Debut
```

```
// DECLARATION  
variable : type
```

```
// INITIALISATION  
variable = 0
```

```
// CORPS DU PROGRAMMES  
instruction 1  
instruction 2  
...
```

Autres actions de base

- lecture : à partir du clavier / fichier
- écriture : écran / sur un support externe
- concaténer : associer variable avec texte

Choix ou Structures Conditionnelles

On a le choix entre plusieurs actions :

- structures conditionnelles pures

ex: SI condition

ALORS

action 1

FIN SI

- structures conditionnelles alternatives

ex: SI condition

ALORS

action 1

SINON

action 2

FIN SI

- structures conditionnelles alternatives imbriquées

ex: SI condition

ALORS

action 1

SINON

SI cond

ALORS

action 2

SINON

action 3

FIN SI

FIN SI

Répétition

Sert à repeter n fois une action :

- TANT QUE
- REPETER ... JUSQU'A : effectue l'action une fois avant de vérifier la condition
- POUR

Tableaux à une dimension

Ils peuvent être préremplis.

L'indexation commence à 0.

Un tableau de N case aura pour indice i de 0 à N-1.

Tableaux à deux dimensions

Lorsqu'un traitement utilise plusieurs tableaux à une dimension, on utilisera un tableau à deux dimensions.

On définit $T(i,j)$ avec i lignes et j colonnes.

Pour lire un tableau à 2 dimensions on doit imbriquer deux boucles l'une dans l'autre :

ex: POUR ligne DE 0 A N-1

POUR colonne DE 0 A M-1

AFF ligne, colonne, $T(\text{ligne}, \text{colonne})$

FIN POUR

FIN POUR

Les Concepts de l'Approche par Objets

3 concepts nécessaires à la POO :

- encapsulage
- héritage
- polymorphisme

Classe

Une classe est un modèle qui décrit une abstraction.

Permet de recenser une série d'éléments communs décrivant un concept précis.

Elles décrivent les propriétés des objets.

Analogie avec plan d'un pavillon, avec chaque pavillon qui peut avoir une couleur différente (plan = classe, pavillon = objet).

Elles sont de 2 types :

- propriétés contenant de l'information :
 - o modifiés au travers de la méthode
 - o ces propriétés sont les "attributs"
- propriétés actives
 - o fournissent un service et peuvent modifier l'état d'un Objet
 - o ses propriétés sont les "méthodes"

ex: classe compte

propriétés (attributs) : solde, decouvertAutorise, dateOuverture

propriétés actives (méthode) : crediter(), debiter(), obtenirSolde(), calculAgios()

Il est possible de conférer un type à un attribut ou à un argument d'une méthode.

Un type désigne un ensemble de valeurs que peut prendre un attribut ou un argument : entier, chaîne, booléen, réel...

Une classe peut également servir de type.

Objet

C'est une matérialisation de la classe (concrétisation).

L'acte de concrétiser une classe s'appelle le mécanisme d'instanciation.

On dit qu'on instancie une classe ou que l'on crée un Objet.

Objet = instance de Classe.

ex: MaVoiture:Automobile

marque = "Peugeot"

modèle = "607"

puissance = 10

couleur = "rouge"

Encapsulation

Pour chaque attribut et ou méthode d'une classe, on précisera son niveau de visibilité.

C'est l'idée de masquer le fonctionnement exacte du mécanisme bien que je puisse utiliser ce mécanisme (idée de la grosse boîte noire)

public	+	élément visible par tous
protégé	#	élément visible par les sous-classes de la classe
privé	-	élément visible seulement par la classe
paquetage	~	élément encapsulé visible uniquement dans les classes du paquetage

ex:

Personne | | => classe

-nom : chaîne | |

-prenom : chaîne | | => attributs

-dateNaissance : Date | |

+calculeAge() : entier | |

+getNom() : entier | | => méthode

+setDateNaissance(dateNaissance : Date) | |

+getNom() est un accesseur en lecture/getter

+setDateNaissance est un accesseur en écriture/mutateur/setter

Propriétés de Classes

Un attribut de classe est partagée avec sa valeur par l'ensemble des instances.

Une méthode de classe est une méthode qui est directement liée à la classe elle-même.

Au sein d'une telle méthode, seuls les attributs dits "de classe" sont accessibles.
Un attribut de classe est un attribut transverse partagé par toutes les instances (ex: TVA pour des produits en ventes, si ils ont la même TVA).

ex: reprenant la classe Personne. l'attribut -majorite : int = 18. On la met en valeur en la soulignant.

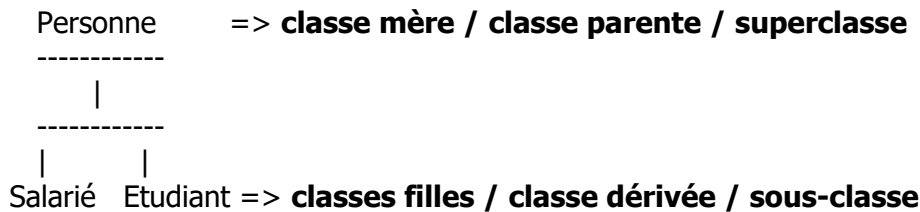
ex: en reprenant la classe Perssonne, une méthode de classe +changeMajorite(nouvelMajorite : entier)

Généralisation/Spécialisation

La généralisation/spécialisation est une association particulière.
Elle ne porte pas sur les instances mais sur les classes.
Elle exprime que les instances d'une classe (classe fille) sont également des instances d'une autre classe (classe mère).

La généralisation a pour but de factoriser des attributs et des méthodes communs à plusieurs classes.

ex: on peut créer une classe spécialisé de personne : Salarié avec -salaire. Une autre Etudiant avec -note et -niveau.



Plus on remonte : généralisation. Plus on descent spécialisation.

On peut créer des classes stériles qui ne peuvent avoir d'enfants.

Héritage

Les instances d'une sous-classe sont aussi instance de ses surclasses.
Ces dernières sont donc également décrites par les attributs et méthodes introduites dans les surclasses.

Par conséquent, une sous-classe hérite des attributs et des méthodes de sa sur-classe.
Elle n'hérite que attributs et méthodes publiques (ou protégés : #) mais pas les publiques.

Cet héritage provient de la relation Généralisation/Spécialisation.

Polymorphisme

Le polymorphisme s'inscrit dans une logique de Généralisation/Spécialisation.

Le polymorphisme est le mécanisme qui consiste à appeler la méthode en fonction du type de l'objet instancié et non pas du type de l'objet réellement déclaré.

ex:

```
Salarié sal1;
sal1.affiche(); => va chercher la méthode affiche() de la classe Salarié
```

```
Personne p1;
p1.affiche(); => va chercher la méthode affiche() de la classe Personne
```

```
Personne TPers[3];
tPers[0] <- p1;    => possible car p1 est une Personne
tPers[1] <- sal1;  => possible car sal1 est une Personne
tPers[0].affiche(); => va chercher la méthode affiche() de la classe Personne
tPers[1].affiche(); => va chercher la méthode affiche() de la classe Salarié car on appelle le type de l'objet
instancié (Salarié) par réellement déclaré (Personne)
```

Classe abstraite

Une classe "concrète" est une classe instanciable car elle décrit un modèle complet, tous les attributs et toutes

les méthodes sont totalement décrits.

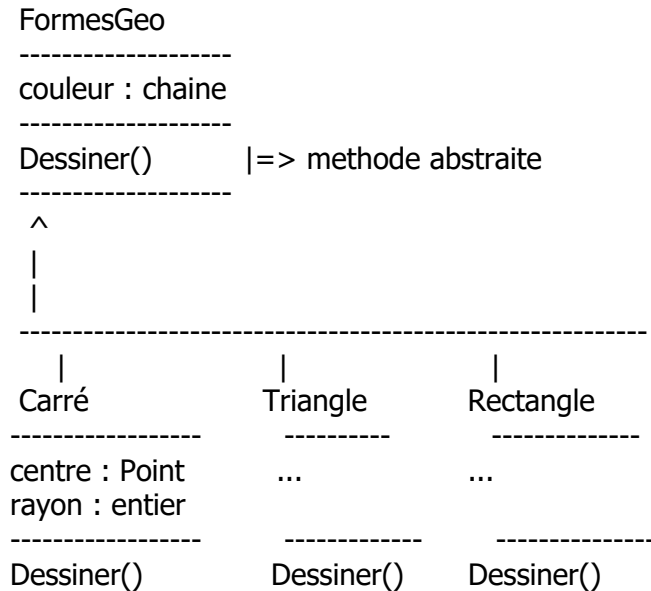
Une classe abstraite provient de la "Généralisation".

Il s'agit d'une factorisation de propriétés communes.

Une classe abstraite ne peut avoir d'instances directes.

Les méthodes d'une classe abstraite peuvent être décrites de façon limitée à la signature de méthodes. On parle de "Méthode Abstraites".

ex:



Traditionnellement, on indique une classe abstraite en l'écrivant en italique.

Ici la classe FormesGeo est abstraite (correspond à rien), mais permet de factoriser l'attribut couleur.

Forcer à avoir une méthode abstraite (donc vide) force à avoir la méthode Dessiner() défini dans les sous-classe.

On garanti le **polymorphisme**.

On impose un "schéma".

Si la sous-classe est elle aussi abstraite il n'est pas nécessaire de définir les méthode abstraite de la surclasse.

Interface

Une interface est une classe abstraite ne contenant que des signatures de méthode publiques.

Une interface est réalisée par une classe.

On dit qu'une classe implémente une interface.

Une interface est employée pour décrire les fonctionnalités d'une classe ou d'un composant.

C'est un élément fondamental de la conception objet.

On indique formellement qu'une classe implémente une interface avec une flèche en pointillées ou le formalisme lollipop Classe Interface ---O)--- Classe

Ligne de Commande

Prof. : ASSOUS Steeve

Code : 2518

Sommaire

- Présentation de Linux/Unix
- Système de fichiers : l'arborescence
- Manipulation de fichiers
- Edition de fichiers texte : VI
- Redirections et Filtres
- Recherche de Fichiers via "find"
- Recherche de motifs "grep"
- Gestion des processus

Présentation

Définition

C'est un OS

Programme ou ensemble de programmes et d'API, interface matériel et applications.

Disponible pour de simple micro (PC, MAC, Atarie...) jusqu' au gros syst_me (IBM Z Series) et même dans des PDA

Unix ? Linux ? Telle est la question.

Historique de UNIX

1969 : Bell Laboratoire; Ken Thompson travaille sur MULTICS; Bell Lab abandonne le projet
K. Thomspon renomme UNICS et dev. sur DEC PDP-7

1974 : refonte du syst. en langage C; Unix gagne la fav. des univ.

1978 : V7 annoncée

1979 : cout des licences incite Berkley à continuer ses travaux. Création de BSD; le Darpa utilise BSD Unix

1983 : AT&T met en vente la version commerciale de Unix Syst. V

1984 : Wenix 1er Unix sur PC

1991 : Apparition des premiers clones Unix comme Linux et FreeBSD

1992 : Sun sort Solaris

D'où vient Linux ?

1985 : création de la FSF par R. Stallman

Copyleft et GPL avec 4 libertés fondamentales :

exec. le programme pour tout usage commercial ou non et par n'importe quel type de personne ou d'organisation

étudier le fonctionnement du programme et l'adapter

redistribuer des copies gratuitement ou non

améliorer le programme et publier les améliorations

1991 : création de Linux par Linus Torvald

Caractéristiques

Unix/Linux est un OS :

- multi-taches
- multi-utilisateurs
- multiplateformes
- stable

Les composants de base d'un Unix/Linux sont le noyau (kernel) et les outils (shell et commandes).

L'OS a pour principales tâches les points suivants :

- gestion de la mémoire
- accès aux périphériques
- accès disque/ système de fichiers
- gestion des programmes (processus)
- sécurité / accès aux données
- collecte d'information système : statistiques

LAMP pour les serveurs : Linux (OS) Apache (serveur Web) MySQL (SGBDR) PHP (langage de programmation serveur).

Quelle distribution ?

Le choix d'une distribution dépend de :

- du coût
- de la suite logicielle
- de la compatibilité matérielle
- des outils d'administration
- mais aussi les préférences de chacun

Les distributions "grands publics"

- Fedora (distribué par Red Hat)
- Open Suse
- Mandriva (anciennement Mandrake)
- Ubuntu

Les distributions "pro."

- Red Hat
- Suse
- Debian
- Slackware

Les distributions "mobiles"

- Knoppix
- Mandriva Flash
- Fedora Live CD

Les distributions "spécialisées"

- Tomsrbt
- µLinux

Connexion : consoles et terminaux

Chaque utilisateur sera doté d'un login et d'un MdP

- root
- règle de mot de passe

La connexion peut se faire de différentes manières

- console virtuelle
 - Texte
 - Graphique
- émulateur de terminaux
- client ssh

Prompt (Invite shell)

Le shell

- interprétation en ligne des commandes
- il traduit les requêtes en actions

Prompt : PS1

- il s'agit de l'invite présente au moment de la saisie d'une commande
- cette invite diffère suivant les environnements et suivant les utilisateurs

ajc1@DebServ:~\$

- ajc1 : username
- DebServ : hostname
- ~ : repertoire de connexion (ici HOME)

La Documentation

- man
- info
- option --help
- internet :
 - www.linux.org
 - www.tldp.org
 - www.redhat.fr
- groupe de discussion

Deconnexion

Trois manières de se déconnecter d'un terminal texte :

- exit
- logout
- ...

Quelques commandes

Info utilisateur

- who
- whoami
- passwd

Affichage

- clear
- echo

Temps

- date
- cal

Système de Fichiers : Arborescence

Organisation

Un syst. de fichiers (FileSystem) définit comment sont gérés les fichiers par l'OS

Présenté de façon arborescente, le FS est une hiérarchie de répertoires ayant pour racine unique "/"

Organisation logique et indépendante du stockage physique des données.
Complètement transparente pour l'utilisateur.

Diff. rep. :

- bin : contient les commandes
- boot : gere le demarage du système
- cdrom : pour monter le cdrom
- dev : contient les périphériques
- etc : contient les fichiers de config
- home : repertoire des utilisateurs
- lib : contient les librairies
- lost+found : garde les fichiers non correctement fermés en cas de crash

- root : rep. du root
- /sbin : commande admin
- tmp : fichiers temporaire
- mnt : montage fichiers
- proc : syst. de fichiers virtuel représentant l'état actuel du syst.
- sys : syst. de fichier vituels représentant l'état des différents peripheriques.
- var : fichiers dont la taille vaire au cours de la vie du systeme
- ...

"Tout est Fichier" : nomenclature

Unix fait la distinction entre min et maj.

La plupart des caracs. (chiffres, lettres, maj. min. certains signes, caractères accentués) sont acceptés, y compris l'espace (déconseillé)

Adressage Absolut/Relatif

Absolu : chemin identifié à partir de la racine

Relatif : chemin défini à partir du répertoire courant

Rep. :

- courant : .
- parent : ..
- personnel : appelé aussi répertoire HOME
→ définit le rep. de connexion, l'espace personnel de l'utilisateur (/home/nomUser)

Manipulation de Fichiers/Répertoires

Commandes permettant de manipuler les fichiers

- pwd : print working directory
- cd : change directory
- ls : liste le contenu du repertoire
→ ls -r : recursif
→ ls -l : affiche sur une seule colonne
- file : renvoie le type de fichier

Répertoire

- mkdir
- rmdir

Fichiers

- touch
- cp
- rm
- mv
- cat
→ cat > nomFichier : permet d'ecrire dans le fichier (attention efface ce qui est écrit)
→ cat >> nomFichier : écrit à la fin du fichier
- more
- head/tail : affiche les 10 premières/dernières lignes du fichier
→ head/tail -x : affiche les x premières/dernières lignes
- cut : affiche une partie des lignes du/des fichier(s) en argument
→ La selection peut se faire par champ ou par caractères
→ cut -d: (délimiteur ici :) -f3 (champ ici 3)
→ cut -c1-8 (caractères de 1 a 8)
- wc : comptabilise le nombre de lignes, mots, caractères

ex :

more /etc/passwd : affiche tous les utilisateurs inscrits

ajc1:x:1151:1003::/home/ajc1:/bin/bash

nomUser:PrésencePasseword:uid:gid:NomGECOS:RepertoireDeCMX:PossibilitéDeSeConnecter

drwxr-xr-x 4 ajc1 unix 4096 oct 5 12:13 dos

- d : type (d : directory, - : fichier ordinaire, l : lien symbolique, b: block, c : caractere)
- rwxr-xr-x : permission User|Group|Other
- 4 : nombre de lien physique
- ajc1 : user propriétaire
- unix : groupe propriétaire
- 4096
- oct 5 12:13 : horodatage
- dos : nom

ls -l | cut -c1-10,41- : liste les fichiers avec détails | en ne gardant que les 10 premiers caractères puis les caractères à partir du 41e.

wc fichHello : affiche le nombre de lignes, mots et caractères dans fichHello

2 5 27 fichHello : 2 lignes, 5 mots, 27 caractères

wc -l /etc/passwd : affiche le nombre de ligne dans /etc/passwd, correspond au nombres d'utilisateurs déclarés.

Permission

rwX :

- r : readable
- w : writable
- x : executable

Pour un fichier normal :

- r : contenu peut être lu, chargé en mémoire, visualisé, recopié
- w : le contenu du fichier peut être modifié, on peut écrire dedans. La suppression n'est pas forcément liée à ce droit (voir droit rep.)
- x : le fichier peut être exécuté depuis la ligne de commande, s'il s'agit soit d'un programme binaire (compilé), soit d'un script

Pour un repertoire

- r :
- w : droit de suppression
- x : droit d'accès

chmod : modifie les permissions

- chmod u+x : ajoute le droit d'exécuter à l'utilisateur
- chmod g-w : retire au groupe le droit de lire
- chmod og+x : ajoute au autres et au groupe le droit d'exécuter
- chmod u=rwx : donne à l'utilisateur le droit de lire, écrire et exécuter
- chmod a-x : retire à tout le monde le droit d'exécuter

chmod peut aussi être utiliser avec une notation octale :

- r : 4
- w : 2
- x : 1

vi

Editeur WYSIWYG

Grand classique des editeurs de texte en entreprise

vi : mode de fonctionnement

Jongler entre les différents modes :

- insertion : i, I, a, A, o, O
- commande : echap
- ligne : `:`

Pour sauvegarder `:wq`

Mode commande

- nG : emmène à la ligne n (identique :n)

vi : Correction

- x : efface le caractère sous le curseur
- X : efface le caractère devant le curseur
- re : remplace le caractère sous le curseur par le 'e'
- dw : efface le mot depuis le curseur jusqu'à la fin du mot
- d\$: (ou D) efface tous les caractères jusqu'à la fin de la ligne
- dd : efface la ligne active
- ndd : efface les n-lignes
- y/Y : copie la ligne
- nY : copie n-lignes
- p/P ; copie en dessous/au dessus de la ligne contenant le curseur

vi : mode ligne

- :w : enregistre
- :1,10w Nom_fich : enregistre les 10 premières lignes
- :r Nom_Fich : insère le fichier Nom_fich à partir de la ligne courante
- :! commande : exécute la commande puis retourne à l'éditeur
- :r ! commande : exécute la commande et inscrit le résultat dans l'éditeur
- :f Nom_fic : affiche en bas de l'écran le nom du fichier, nb de ligne et position actuel

vi : recherche et substitution

Recherche :

- on distingue deux façons de rechercher des chaînes dans vi
 - / : recherche de haut en bas
 - ? : recherche de bas en haut

Substitution :

- :[1ère ligne, dernière ligne]s/Modèle/Remplacement/[gil]
 - :1,5/bin/sbin/
 - :5,30s/bach/ksh/g
 - 1,\$s/fonction/function/gi
 - \$ (ou %) jusqu'à la fin du fichier
 - g : toutes les occurrences
 - i : insensible à la casse
 - I : sensible à la casse

vi : options

Options de l'éditeur

- :set all : liste l'ensemble des options disponibles
- :set number/nonumber : affiche/cache les numéros de lignes
- :set autoindent/noautoindent : l'indentation est conservée lors d'un retour de ligne
- :set showmode/noshowmode : vi affiche une ligne d'état (montre le mode actuel)
- :set tabstop=x : définit le nombre de caractères pour une tabulation
- :set showmatch/noshowmatch : montre la parenthèse d'ouverture lors de la fermeture de cette dernière
- ab DF DUPONT Franck : crée une abbréviation
- map x 10dd : mappage des touches

Pour conserver les options il faut créer un fichier spécifique avec les options (.exrc).

Commentaires avec "

Redirections et filtres

Tout processus lancé est associé à 3 descripteurs (input, output, error)

ex: ls

- entrée standard : dossier courant
- sortie standard : écran

mkdir rep

- entrée standard : rep
- sortie standard : écran

cat > f1

- entrée standard saisie redirigée en sortie vers f1

'>' est un opérateur de redirection de même que '>>' (mais lui redirige à la fin du fichier).

Opérateurs de filtre (pipe)

- | : permet de lier des commandes. La 2eme commande est liée au résultat de la 1ere.

Find : prototype

Permet de rechercher des fichiers

Find : critères de recherche

au niveau du nom :

- -name | iname motif
- ex: \$find . -name 'nom'

au niveau du temps :

- -mtime [+ -] n (n jours) : modif de contenu
- -atime [+ -] n : accéder
- -ctime [+ -] n : modif de contenu et/ou de caractéristiques
- ex: \$find . -atime -1

au niveau des caractéristiques :

- -size [+ -] taille [bck]
- -user utilisateur
- -group groupe
- -type d|f|l|b|c : dossier|files|link|
- -perm droits
- ex : \$ find / -user 'as1'

Combiner des critères

- et logique : -a ou -and
- ou logique : -o ou -or
- negation : -not ou !

Exécution des commandes

- -print : affiche le nom des occurrences trouvés
- -exec : exécute la commande spécifiée
- -ok : reprend l'option -exec mais avec une demande de confirmation
- -ls : affichage des résultats

ex:

- find . \(\(-name 'D*' -type d \) -o \(-name 'rep*' -type d \) \) -print
- find . -name '*.sh' -exec cat {} \; : applique cat à chaque résultat du find.

grep

grep : prototype

Extraire des lignes d'un fichier selon divers critères.

Pour cela on dispose de trois commandes grep, egrep et fgrep qui lisent les données soit depuis un fichier d'entrée soit depuis le canal d'entrée standard.

grep [-options] motif [arguments]

Quelques options :

- -v : affiche toutes les lignes qui ne correspondent pas au motif
- -c : ne retourne pas les lignes mais leur nombres
- -i : ne tient pas compte de la casse
- -n : indique le numéro de ligne pour chaque ligne trouvée
- -l : affiche le nom du fichier

grep : Expressions régulières

Quelques métacaractères

- ^ : début de ligne
- . : un caractère quelconque
- \$: fin de ligne
- x* : zéro ou plus d'occurrences du caractère x
- x+ : une ou plus occurrences du caractère x
- x? : 0 ou une occurrence unique de x
- [aX0] : classe de caractères permis
- [.-.] : plage de caractères permis
- [^...] : plage de caractères interdits
- \{n,m\} : facteur d'occurrences
- \ : caractère d'échappement
- () : définit un groupe
- | : ou

ex:

- grep '^unix' /etc/passwd : commence par unix
- grep '[^0-9]\$/etc/passwd : ne se termine pas par un chiffre
- grep '^t(iti|oto|ata)' nomFichier : commence par titi toto tata
- grep ^0[1-9](\[0-9\]\{2\})\{4\} : recherche un numéro de téléphone 0x.yy.yy.yy.yy avec x 1-9 et y 0-9*

Gestion des processus

Un processus représente à la fois un programme en cours d'exécution et tout son environnement d'exécution (mémoire, état, ..)

Définit par :

- un numéro de processus unique PID
- un numéro de processus parent PPID
- un numéro d'utilisateur et un numéro de groupe
- un répertoire de travail

Daemons

La commande ps affiche les processus en cours (ou pstree).

- liste les processus du syst.
- liste les processus de façon hiérarchique

Quelques options :

- a : tous les utilisateurs
- u : nom des utilisateurs et date
- x : non rattaché à un terminal
- l : informations détaillées

Quelques colonnes :

- UID
- PID
- PPID
- C : facteur de priorité (plus la valeur est grande plus la priorité est basse)
-

Signaux et commande "kill"

Le signal est l'un des moyens de communications entre les processus.

Lorsqu'on envoie un signal à un processus, celui-ci doit l'intercepter et réagir en fonction de celui-ci.

Certain signaux peuvent être ignorés, d'autres non.

Suivant les UNIX; on dispose d'un nombre plus ou moins important de signaux.

L'option -l permet d'obtenir la liste des signaux.

Liste signaux :

- 1 SIGHUP : hang up, est envoyé par le père à tous ses enfants
- 5 SIGTRAP :
- 2 SIGINT : interruption du processus demandé (CTL+C)
- 3 SIGQUIT : idem SIGINT mais génération d'un core dump (fichier débogage)
- 9 SIGKILL : signal ne pouvant être ignoré, force le processus à finir brutalement
- 15 SIGTERM : signal envoyé par défaut par la commande kill. Demande au processus de se terminer normalement

Manipulation processus

Lancer un processus :

- en avant-plan : on attend la fin de la commande avant de récupérer la main.
- en arrière-plan : la commande se lance mais on ne perd pas la main. On utilise l'esperluette &.
- persistant : commande "nohup"
- priorité : nice/renice

Arrêter un processus :

- Ctrl + C : signal SIGINT
- Ctrl + \ : signal SIGQUIT
- Ctrl + Z : signal SIGSTOP

Contrôler les processus

- commande "jobs" : affiche la liste des processus lancés en arrière plan
- commande "bg" : relance un processus suspendu en arrière plan
- commande "fg" : lance un processus en avant-plan

Shell

Prof. : ASSOUS Steeve

Code : 4826

Sommaire :

Qu'est-ce-que le shell ?

Le shell est un interpréteur de commandes :

- initialise l'environnement
- génère le prompt
- assigne les variables
- effectue les substitutions de variables et commandes
- interprète les métacaractères
- génère les noms de fichiers
- gère les redirections et filtres
- gère l'historique des commandes
- complétion de noms de fichiers
- création d'alias

Différents types de shell :

- Ksh : Korn Shell
- Sh : Bourne shell
- Csh
- Bash : Bourn again shell

Que se passe-t-il lors de la connexion ?

Envoie de l'invite de connexion

- saisie des informations de connexion

Effecture la connexion

- validation du login
- place l'utilisateur dans son repertoire de connexion
- lance l'interpréteur de commandes

Etapas de l'interpréteur de commandes

- execute le fichier `"/etc/profile"`
- execute le fichier `"~/ .profile"`
- execute le fichier `".bashrc"` (fichier bash utilisé)
- interpréteur de commande

umask fonctionne à l'inverse du chmode :

	Repertoire	Fichier
0	rwX	rw-
1	rw-	rw-
2	r-X	r--
3	r--	r--
4	-wX	-w-
5	-w-	-w-
6	--X	---
7	---	---

Substitution de variables

Une variable

- est un nom associé à une valeur
- **n'est valable que dans le shell où elle a été défini**
- doit commencer par une lettre, et ne contenir que des lettres, chiffres et underscore

Affectation `var = "toto"`

Affichage `echo = $var`

Notion d'Alias : "alias"

un alias est une substitution d'une commande par un raccourci

L'alias est prioritaire sur les fonctions, commandes

On peut rendre un alias permanent en le rajoutant au `.bashrc`.

On le supprime avec : `unalias [-a] name [name ...]`

Liste des variables : "set"

Permet d'afficher les noms et valeurs de toutes les variables du shell en zone de données et dans l'environnement.

`unset nomvariable`

- réinitialise le contenu d'une variable

`unset`

- réinitialise toutes les variables.

Variables d'environnement : "env"

Permet de lister les variables de l'environnement

Certaines variables décrivent des informations liées au système

- HOME
- PATH : définit les chemins à scruter lors de l'exécution des commandes
- PS1 : prompt primaire
- PS2 : prompt secondaire
- TERM : le type de terminal
- PWD : le dernier repertoire de travail
- LOGNAME : le nom de l'utilisateur logué

Export de variables

Par défaut une variable a une portée locale et n'est donc accessible uniquement dans le shell où elle a été défini.

La commande "export" permet de promouvoir les variables en tant que variables d'environnement et les rend

accessibles dans l'environnement utilisateur
- export nomVariable

Rendre un fichier executable

Creer un script
- vi nomScript.sh ...

Le rendre executable
- chmod 755 nomScript.sh

Execution
- ./nomScript.sh

Pour rendre le chemin disponible dans le PATH, il faut le rajouter, par exemple dans .bashrc ou .profile.
Ne pas oublier de faire l'export.

Substitution de commande

Deux façons :
1. jour=\$(date)
2. jour=`date`

Metacaractères

- # : commentaire
- \$: substitution de variable
- & : lancer en arrière-plan
- && : enchainement conditionnel
- || : enchainement conditionnel
- | : filtre
- > : redirige la sortie standard
- 2> : redirige la sortie erreur
- ' : délimite une chaine de caractères. Les caractères spéciaux sauf \ et ` perdent leur signification
- " : délimite une chaine de caractères. Les caractères spéciaux sauf \, \$ et ` perdent leur signification
- `cmd` : Execution et substitution de commandes
- \$(cmd) : Execution et substitution de commandes
- \ : caractère d'échappement
- >> : redirection de la sortie standard avec ajout.

Passage d'argument

Les paramètres positionnels

- ◇ Le nom du fichier et les paramètres de la ligne de commande sont stockés et peuvent être récupérés au sein d'un shell script
 - \$0 : nom de domaine (du script)
 - \$1-9 : les neufs premiers paramètres du script
 - \$# : nombre total de paramètres
 - \$* : Liste des arguments
 - \$@ :

Variables spéciales

- ◇ \$? : code retour de la dernière commande exécutée (0 réussi, 1 echec)
- ◇ \$\$: PID du shell actif
- ◇ \$! : PID du dernier processus lancé en arrière plan
- ◇ \$- : options du shell

exit x : permet de sortir avec un code retour de x (0-255)

- ◇ théoriquement :
 - 0 : si tout se passe bien
 - 1 : si erreur dans la commande

- 2 : si commande fausse
- **Attention ! Il y a de nombreux contres exemples.**

Commande "shift"

Syntaxe :
shift [n]

But :

- ◇ décale de n arguments vers la gauche
- ◇ n prend 1 par défaut
- ◇ met à jour la variable \$#

Saisie à l'écran "read"

Permet la saisie de chaîne de caractère à partir du clavier.

Options :

- -p : read -p "Saisir un nom" nom, affiche une phrase
- -e : echo -e "Nom Saisi \n" \$echo

Double expansion : "eval"

"eval" construit une commande en concaténant ses arguments.

Le shell lit et exécute cette commande.

Commande	Affichage
var = 10	10
x=var	var
y=\\$x	\$var
echo \$y	\$var
eval echo \$y	10

Opérations sur les entiers : "expr"

Permet l'arithmétique entière sur les valeurs : + - / % *

Role et Comportement du Consultant

Red Hat System Administration

Installation et Configuration de Microsoft Windows Server 2012

Savoir se Présenter, les Nouvelles Compétences Acquises

PowerShell

Programmation Python

L'Ingenieurie DevOps sur Amazon Web Services

Oracle SQL et Exploitation

Docker

Jenkins

Ansible

Puppet

Tips

Divers

Shell

- `\n` : retour à la ligne
- `\t` : tabulation horizontale
- `\v` : tabulation verticales