

Parcours de formation SIN

**Déploiement d’une interface de communication web**

Module SIN412 : COMMANDES LINUX ET PROGRAMMATION C

SOUS LINUX

Durée : 3h

Objectifs à atteindre : Acquérir la connaissance des commandes fondamentales des systèmes d'exploitation Unix et Linux.

Niveau des connaissances envisageable : BAC

Pré requis : - Maitrise de l’environnement et du système de fichier de linux Ubuntu.

- Le langage C

Logiciels utilisés : - VMware Workstation et/ou Player.

- Dernière version stable de Linux Ubuntu

- Putty

- gcc

Outils mobilisés, le cas échéant : tous les tutoriels disponibles sur internet

Webographie :

<http://www.siteduzero.com/tutoriel-2-96-linux.html> (*Jarvis, licence CC*)

<http://www.ubuntu-beginner.com/>

<http://doc.ubuntu-fr.org/terminal>

<http://www.linux-france.org/article/debutant/debutant-linux.html>

<http://doc.ubuntu-fr.org/tutoriel/console_ligne_de_commande>

<http://doc.ubuntu-fr.org/tutoriel/script_shell>

<http://doc.ubuntu-fr.org/console>

Illustrations :

Copies d’écran Linux Ubuntu – Terminal – M. Silanus.

Sommaire

[1 Les principales commandes linux 3](#_Toc288953844)

[1.1 Le shell 3](#_Toc288953845)

[1.2 Le terminal *shell* 3](#_Toc288953846)

[1.2.1 Qu’est-ce qu’une commande ? 4](#_Toc288953847)

[1.2.2 Format d’une commande 4](#_Toc288953848)

[1.2.3 Facilités de saisie des commandes 4](#_Toc288953849)

[1.3 Les commandes fondamentales 5](#_Toc288953850)

[1.4 Les utilisateurs et les groupes 6](#_Toc288953851)

[1.5 Les redirections 7](#_Toc288953852)

[2 Le système de fichier 8](#_Toc288953853)

[2.1 Organisation 8](#_Toc288953854)

[2.2 Les droits d’accès 9](#_Toc288953855)

[3 Les commandes réseau 10](#_Toc288953856)

[4 Exercices 11](#_Toc288953857)

[4.1 Manipuler un dossier ou un fichier 11](#_Toc288953858)

[4.2 Modification des droits d’accès 12](#_Toc288953859)

[4.3 Rechercher un fichier ou un dossier 12](#_Toc288953860)

[4.4 Rechercher un mot dans un fichier 12](#_Toc288953861)

[4.5 Suppression d’un dossier non vide 12](#_Toc288953862)

[5 Programmation en C sous linux 13](#_Toc288953863)

[5.1 Premier programme, compilation et exécution 13](#_Toc288953864)

[5.2 Exercices 13](#_Toc288953865)

[5.3 Elements de correction 14](#_Toc288953866)

# Les principales commandes linux

## Le shell

Un interpréteur de commandes (le « *shell* », la coquille qui entoure le « noyau » du système) est un programme qui sert d'intermédiaire entre l'utilisateur et le système d'exploitation.

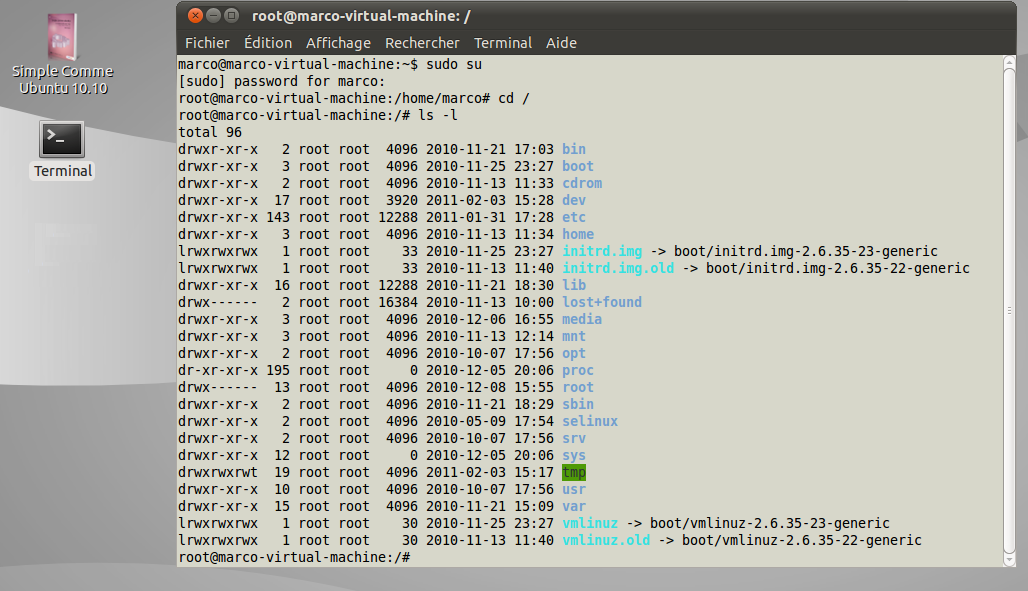
Sa tâche essentielle est l'exécution de programmes.

Pour cela, il effectue (en boucle infinie) :

* la lecture d'une ligne,
* sa compréhension comme une demande d'exécution d'un programme avec d'éventuels paramètres,
* le lancement de ce programme avec passage des paramètres,
* d'éventuelles redirections d'entrées-sorties,
* les exécutions de scripts (fichiers de commandes).

## Le terminal *shell*

Le terminal *shell* est une interface homme-machine permettant à l’utilisateur de saisir des commandes *shell* pour administrer son ordinateur sans utiliser d’interface graphique.



* **Ouvrez le terminal de linux Ubuntu.**

Il est possible de prendre la main à distance sur un système linux à l’aide du protocole *ssh* (*secure Shell*) qui permet d’échanger des données de manière cryptées entre deux ordinateurs. La session commence donc par un échange de clés de cryptage.  
Pour cela, on utilise la commande ssh suivie de l’adresse IP du poste à connecter, par exemple pour ouvrir une session distante sur le poste d’adresse 192.168.1.1, on saisirait la commande :

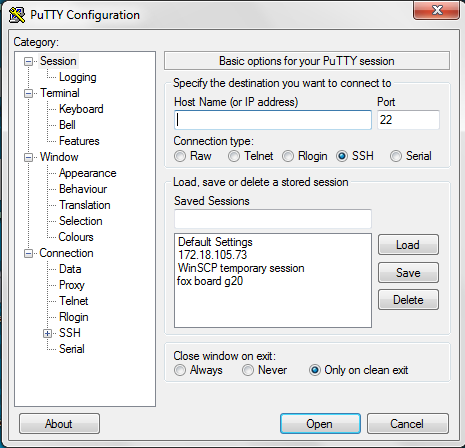
|  |
| --- |
| # ssh 192.168.1.1 |

Si on utilise un ordinateur sous Windows, on peut ouvrir une session distante sur un linux grâce au logiciel putty.exe  ( <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe> ):

Il faudra au préalable avoir installé ssh sur le poste linux distant :

sudo apt-get install ssh

Puis suivez les instructions.



### Qu’est-ce qu’une commande ?

Une commande est un fichier exécutable agissant sur le système ou sur des fichiers. Elles se trouvent généralement dans le dossier /bin.

Chaque utilisateur peut personnaliser sa liste de commande. La variable système PATH contient une liste de dossiers qui contiennent les commandes accessibles.

**Normalement, seul l’utilisateur root (super-utilisateur ou administrateur) a accès à toutes les commandes. (rappel : utilisez la commande sudo su pour devenir root).**

La commande whereis permet de connaitre l’emplacement d’une commande et plus généralement d’un fichier.

* **Recherchez les dossiers qui contiennent la commande ls : A l’invite de commande saisissez  whereis ls**
* **Recherchez les dossiers qui contiennent la commande cat**
* **Identifiez les chemins d’accès aux commandes qui vous sont accessibles : saisissez echo $PATH**

### Format d’une commande

Une commande est constituée d’au moins un mot : le nom de la commande. D’autres éléments sont susceptibles de la préciser. Les mots constituant la commande sont séparés par un espace. Certain caractères ont une signification spéciale. On les appelle méta-caractères.

**nom\_cmd [ option ... ] [ argument ... ]**

Par exemple :

|  |
| --- |
| ls –l /home/m\*   * ls : nom de la commande qui Affiche les informations des fichiers * -l : option * /home/m\* : argument * \* : méta-caractère. Remplace une ou plusieurs lettres |

Pour obtenir la syntaxe et le format précis d’une commande, utilisez comme option --help.

* **Saisissez la commande ls --help et cherchez les options nécessaires pour afficher en format long la liste des fichiers contenu dans le dossier personnel de l’utilisateur courant (/home/nom\_utilisateur), y compris les fichiers cachés (qui commencent par « . » ex : .texte.txt )**

### Facilités de saisie des commandes

Comme les commandes Unix sont souvent longues à saisir, diverses facilités sont offertes :

* **Historique des commandes**

Cette liste numérotée est accessible en tapant **history | less** (appuyer sur Q pour quitter l’historique)   
Pour relancer la commande numéro **n**, saisir (sans espace) **!n**

On peut aussi parcourir les précédentes lignes de commandes avec les flèches (comme doskey) et les éditer. Ceci permet très facilement de reprendre une précédente commande pour l'éditer et la modifier.

* **Le clic-droit**

Dans un terminal console, sélectionner un texte quelconque. Un clic-droit recopie ce texte sur la ligne de commande, même dans une autre console.

* **L'opérateur tilde**

Le caractère **tilde ~** (alt 126) seul renvoie au dossier personnel de l'utilisateur actuel.

Si l’utilisateur actif est toto, chaque occurrence du caractère ~ est remplacé par le chemin /home/toto

Le tilde ~ suivi d'un nom d'utilisateur, renvoie à son répertoire personnel (ainsi la commande cd ~stage conduit à /home/stage)

* **Compléter une commande**

Lorsqu'on tape une ligne de commande incomplète, puis sur la touche **TAB**, l'interpréteur cherche à compléter, nom du fichier ou nom de commande, suivant le contexte.

S'il y a plusieurs propositions, il y a attente d'un complément d'info de la part de l'utilisateur (parfois avec un « ***tut***»).

Un autre **TAB** et l'interpréteur affiche toutes les possibilités ou en indique le nombre, s'il y en a beaucoup !

Exemple :

|  |
| --- |
| * Saisissez le début de la commande suivante et appuyez sur <TAB>   **[home/toto]$** less  /e <TAB>  **[home/toto]$** less  /etc/fs <TAB>  **[home/toto]$** less  /etc/fstab <Entrée>  **…**  <Q> pour quitter |

## Les commandes fondamentales

|  |  |  |
| --- | --- | --- |
| **Commande** | **Description** | **Exemple d’utilisation** |
| cd | *Change directory* : Se déplacer dans les dossiers. | cd /dossier |
| ls | *Liste* : Lister les fichiers d'un dossier. | ls –al |
| cat | *Concatène* : Affiche le contenu d’un ou plusieurs fichier(s). | cat /etc/passwd |
| cp | *Copy* : Copier un fichier | cp filesrc filedest |
| rm | *Remove* : Supprimer un fichier | rm file  rm –r dossier |
| mkdir | *Make directory* : Créer un répertoire. | mkdir dossier |
| mv | *Move* : Déplacer ou renommer un fichier. | mv filesrc filedest |
| find | *Find* : Retrouver l’emplacement d’un fichier. | find –name file |
| grep | *Global regular expression print* : Recherche dans un ou plusieurs fichiers. | grep expression file |
| df | *Disk free*: Connaître l'espace disque restant | df |
| gzip | *Gnu zip*: compactage et décompactage des fichiers au format .gz | gzip -9 file  gunzip file.gz |
| tar | *Tape archiver :* Archivage de données. | tar -cvf file.tar file  tar -xvf file.tar |
| pstree | *Process status tree*: Affiche l'arbre des processus en cours d'exécution. | pstree |
| ps | *Process status :* Affiche les processus en cours d'exécution | ps –A | grep procname |
| kill | *Kill*: Envoyer un signal à un processus (terminer par défaut) | kill 1234 |
| ifconfig | *Interface configuration :* Configuration des interfaces réseau | ifconfig eth0 auto |

Pour obtenir la syntaxe complète d’une commande, consultez le manuel : man commande

## Les utilisateurs et les groupes

Les utilisateurs ne sont pas forcément des personnes. Le système a besoin d’utilisateurs pour sa gestion interne, notamment comme propriétaire des divers processus.

Les utilisateurs appartiennent à un ou plusieurs groupe(s). Ils ont des droits d’accès aux fichiers et d’utilisation des commandes limités. Seul l’utilisateur root possède tous les droits sur le système. C’est le super utilisateur ou administrateur.

Les principales commandes de gestion des utilisateurs et des groupes sont :

|  |  |
| --- | --- |
| useradd  usermod  userdel  passwd | gestion des comptes utilisateur |
| groupadd  groupmod  groupdel | gestion des groupes |

La commande sudo permet d’élever les droits de l’utilisateur au rang d’administrateur, soit temporairement, soit de manière permanente :

|  |
| --- |
| Elévation permanente des droits :  L’utilisateur en cours devient *root*  Le prompt devient # ($ simple utilisateur)  Le dossier est créé  Elévation temporaire des droits pour exécuter la création du dossier  L’utilisateur marco tente de créer un dossier test dans /home : **Interdit** |

Remarquez la forme du prompt :

|  |
| --- |
| marco@marco-virtual-machine:~$ => Prompt de l’utilisateur *marco* sur *marco-virtual-machine*  => $ : indique que vous avez les droits utilisateurs  => ~ : « indique que le dossier actuel est celui de l’utilisateur  root@marco-virtual-machine:/home/marco# => Prompt de l’utilisateur *root* sur *marco-virtual-machine*  => # : indique que vous avez les droits d’administration  => /home/marco : dossier actuel |

|  |
| --- |
| * **Quelle option de la commande useradd permet de créer le dossier personnel du nouvel utilisateur.** * **Octroyez-vous les droits *root* : tapez sudo su (le mot de passe est celui de l’utilisateur courant).** * **Créez un nouvel utilisateur nommé *stagiaire* et créez son dossier personnel (/home/stagiaire).** * **Créez son mot de passe : passwd stagiaire** * **Créez un dossier *test* dans le dossier personnel de *stagiaire*.** * **Listez avec les détails les fichiers et les dossiers de l’utilisateur *stagiaire*.**     Groupe  Propriétaire  **Constatez que le propriétaire du dossier test est *root* et bien que ce dernier soit placé dans le dossier de stagiaire, celui-ci ne peut pas y écrire. Seul *root* possède tous les droits (rwx : read, write, execution) sur ce dossier.**   * **Déconnectez-vous de *root* : exit** * **En tant que simple utilisateur, tentez de supprimer le dossier *test*.** * **En tant que simple utilisateur, tentez de supprimer l’utilisateur *stagiaire* ainsi que son dossier personnel.** * **Refaite de même en tant que *root*.** * **Pour la suite du travail, créez un second utilisateur (avec le nom de votre choix).** |

## Les redirections

Lorsqu’on exécute une commande, le *shell* affiche le résultat sur l'écran. On peut toutefois rediriger cette sortie vers un fichier en utilisant le signe >.

|  |
| --- |
| Affichage du contenu du fichier *resultat-ls*  La commande créée le fichier *resultat-ls* qui contient son résultat |

Dans l’exemple ci-dessus, si le fichier *resultat-ls* existe déjà, il sera écrasé et recréé. Son contenu précédent est définitivement perdu.

Pour écrire à la suite d’un fichier existant, il faut utiliser le signe >>

|  |
| --- |
|  |

Il est possible de rediriger aussi les entrées depuis un fichier. On utilise pour cela le symbole <.

Les redirections sont très intéressantes pour le dialogue avec les périphériques. En effet, sous linux, les ports séries ou parallèles sont représentés par des fichiers situés dans le dossier */dev*.

Pour écrire sur le port série COM1 (*ttyS0*), il suffira de rediriger le flux de sortie vers le port :

|  |
| --- |
| # echo message à transmettre sur ttyS0 > /dev/ttyS0 |

De même, pour recevoir des données en provenance du port, on redirigera le flux d’entrée depuis le port *ttyS0* :

|  |
| --- |
| # cat < /dev/ttyS0  # message reçu depuis le port série ttyS0 (appuyez sur Ctrl+C pour terminer)  # Ctrl+C |

# Le système de fichier

## Organisation

Le système de fichiers correspond au type de formatage qui va être appliqué à une partition.

Sous Linux, le système de fichiers est organisé à partir d'une arborescence unique. Il fait référence aux disques durs locaux ou distants, lesquels peuvent contenir une ou plusieurs partitions.

Il est compatible avec la plupart des systèmes de fichiers existant dans le monde (*FAT*, *FAT* *32* ou *NTFS* de Microsoft, …) et il peut gérer la plupart des supports physiques.

Le système de fichier généralement utilisé sous linux est *ext4* (*Extended File System V4*). Il est journalisé et possède la faculté de pouvoir stocker des fichiers de très grande taille sans les fragmenter.

La racine du système de fichier de linux est représentée par le symbole /. Ce même symbole permet de séparer les dossiers.

**Attention : pour séparer les dossiers, utilisez**

* Windows : \
* Linux : /

|  |  |
| --- | --- |
|  | * ***bin*** : contient les programmes (exécutables) qui sont susceptibles d'être utilisés par tous les utilisateurs de la machine. * ***boot*** : fichiers de démarrage de Linux. * ***dev*** : fichiers contenant les périphériques. * ***etc*** : fichiers de configuration. * ***home*** : répertoires personnels des utilisateurs. * ***lib*** : dossier contenant les bibliothèques partagées (généralement des fichiers .so) utilisées par les programmes. * ***Media***, ***mnt*** : Accès aux périphériques amovible (SDcard ou une clé USB). * ***root*** : dossier personnel de l'utilisateur *root*. * ***sbin*** : contient des programmes système importants. * ***tmp*** : Stockage temporaire des fichiers. * ***usr*** : Dossier d’installation de la plupart des programmes demandés par l'utilisateur. * ***var*** : Contient des données « variables » (fichiers logs, fichiers de serveur web, …) |

## Les droits d’accès

Sous linux, tout fichier appartient à un propriétaire unique et à un groupe d’utilisateur.

Le propriétaire par défaut d’un fichier est celui qui l’a créé. Toutefois, l’utilisateur *root* peut modifier le propriétaire d’un fichier. On utilise alors la commande chown.

|  |
| --- |
| # chown propriétaire fichier |

Son groupe est par défaut le groupe principal auquel appartient le propriétaire.

Seul le propriétaire du fichier et *root* peuvent changer ses droits d’accès.

Un fichier, comme un dossier, possède trois séries de droits d’accès :

* Ceux du propriétaire,
* Ceux du groupe,
* Ceux de tous les utilisateurs.

Ils sont représentés par les lettres :

* r : (read) droits d’accès en lecture
* w : (write) droit d’accès en écriture
* x : (execution) droits d’accès en exécution.

La commande chmod permet de modifier les droits d’accès d’un fichier. On utilise une représentation numérique octale pour représenter les droits :

|  |
| --- |
| # chmod 777 fichier1 (Attribution de tous les droits à tous le monde)  # chmod 755 fichier2 (Attribution de tous les droits au propriétaire et lecture-exécution aux autres utilisateurs)  # chmod 700 fichier3 (Attribution de tous les droits au propriétaire et aucun aux autres utilisateurs)  # chmod +wx fichier4 (Ajout des droits d’écriture et d’exécution à tous le monde) |

Par convention la présence d'un droit est noté 1, l'absence 0.

                             Binaire ----- Droit ----- Octal   
             000 -------- (---) ------- 0   
             001 -------- (--x) ------- 1   
             010 -------- (-w-) ------- 2   
             011 -------- (-wx) ------- 3   
             100 -------- (r--) ------- 4   
             101 -------- (r-x) ------- 5   
             110 -------- (rw-) ------- 6   
             111 -------- (rwx) ------- 7

|  |
| --- |
| * **En tant que simple utilisateur, créez un dossier dans votre dossier personnel, puis listez en le contenu.** * **Qui est le propriétaire par défaut du dossier créé ? Qui est le groupe par défaut ?** * **Quels sont les droits d’accès du dossier actuel en notation numérique octal ?** * **Reconnectez-vous au système en utilisant un autre utilisateur. Vérifiez que vous ne pouvez pas accéder en écriture au dossier précédemment créé.** * **Modifiez les droits pour que tout le monde puisse avoir un accès complet au dossier. (attention, vous devez être *root* ou le propriétaire du dossier)** * **Testez à nouveau l’accès en écriture du dossier.** * **Attribuez à *root* la propriété du dossier. (attention, vous devez être *root*).** |

# Les commandes réseau

L’administration du réseau sous linux passe par la connaissance d’au moins une commande pour la gestion des interfaces :

La commande ifconfig qui permet d’identifier l’adresse réseau fournie par un serveur DHCP ou de la configurer.

|  |
| --- |
| Affichage des paramètres réseaux de toutes les interfaces  $ ifconfig  Configuration de l’interface eth0 en automatique (DHCP)  $ ifconfig eth0 auto  Configuration manuelle de eth0  # ifconfig eth0 192.168.1.15 netmask 255.255.255.0  Activer ou désactiver l’interface eth0  # ifconfig eth0 up|down |

|  |
| --- |
| * **Quelle est l’adresse IP et le masque de votre interface réseau ?**      * **Désactivez votre interface réseau, puis affichez ses paramètres. Vérifiez l’accès au réseau.**      * **Réactivez-la et vérifiez de nouveau l’accès réseau.** |

# Exercices

## Manipuler un dossier ou un fichier

|  |
| --- |
| Déplacez-vous dans le dossier Documents/ et listez le. Créez un nouveau dossier appelé ExoShell et listez à nouveau le dossier Documents/ .  Commandes utilisées : |

|  |
| --- |
| Déplacez-vous dans le dossier ExoShell et lister son contenu. Essayez de le supprimer tout en restant dedans. Que se passe t-il ?  Commandes utilisées :  Faites en sorte de pouvoir le supprimer.  Commandes utilisées : |

|  |
| --- |
| Créer le fichier texte.txt et insérez-y le texte suivant : (Copier-coller)  *Linux, ou GNU/Linux, est un système d’exploitation compatible POSIX de type UNIX. Linux est fondé sur le noyau Linux, logiciel libre créé en 1991 par Linus Torvalds. Il est souvent utilisé avec les logiciels du système d’exploitation libre GNU.*  *Développé sur Internet par des milliers d’informaticiens bénévoles et salariés, Linux fonctionne maintenant sur du matériel allant du téléphone mobile au superordinateur. Il existe de nombreuses distributions Linux indépendantes, destinées aux ordinateurs personnels et aux serveurs informatiques, pour lesquels Linux est très populaire. Elles incluent des milliers de logiciels issus de la communauté du logiciel libre et fréquemment quelques logiciels propriétaires. Linux est également populaire sur système embarqué. La mascotte de Linux est le manchot Tux.*  Commandes utilisées :  Ajoutez à la fin du fichier la source de cet article : *Linux - définition par Wikipedia*  Commandes utilisées :  Affichez le contenu du fichier.  Commandes utilisées : |

|  |
| --- |
| Créez un dossier appelé texte et copiez-y le fichier texte.txt précédant. Listez le contenu de ce dossier.  Commandes utilisées :  Renommez le fichier texte.txt en linux.txt  Commandes utilisées : |
| Supprimez le fichier texte.txt  Commandes utilisées : |

## Modification des droits d’accès

|  |
| --- |
| Qui est le propriétaire du dossier texte ?  Commandes utilisées :  Attribuez la propriété du dossier texte à *root* et interdisez l’accès à ce dossier à tous les utilisateurs (sauf *root*).  Commandes utilisées :  Qui est le propriétaire du fichier linux.txt contenu dans le dossier texte ?  Commandes utilisées :  En tant qu’utilisateur propriétaire de ce fichier, pouvez-vous lire son contenu ? Pourquoi ?  Commandes utilisées :  Modifiez les droits d’accès nécessaire pour que seuls *root* et le propriétaire du fichier puissent le lire et le modifier.  Commandes utilisées : |

## Rechercher un fichier ou un dossier

|  |
| --- |
| A partir de la racine, chercher l’emplacement du dossier texte  Commandes utilisées :  A partir de la racine, chercher l’emplacement du fichier linux.txt  Commandes utilisées : |

## Rechercher un mot dans un fichier

|  |
| --- |
| Recherchez le mot « linux » dans le fichier linux.txt indépendamment de la casse (majuscules ou minucules). La commande à utiliser est la commande grep. Pour connaitre la manière de l’utiliser, utilisez l’aide :  # grep --help  Commandes utilisées : |

## Suppression d’un dossier non vide

|  |
| --- |
| Supprimez le dossier texte et tout ce qu’il contient en une seule opération.  Commandes utilisées : |

# Programmation en C sous linux

## Premier programme, compilation et exécution

Ouvrez l’éditeur de texte Gedit (saisissez la commande gedit en tant que root). Saisissez le programme suivant et enregistrez le dans votre dossier personnel sous le nom « **helloword.c**»

|  |
| --- |
| #include<stdio.h>  main()  {  printf("Hello World");  } |

Le compilateur «**gcc**» est intégré nativement dans toute distribution de linux. Pour l'utiliser, saisissez la commande suivante dans un terminal :

|  |
| --- |
| **# gcc helloword.c -o executable** |

Si la compilation s'est bien passée, aucun message n'est affiché dans le prompt. Sinon, réglez le problème et recommencez.

Vous disposez maintenant dans votre dossier personnel d'un fichier «**executable**» qui constitue l'exécutable créer par le compilateur.

Pour l'exécuter, saisissez la commande suivante dans un terminal :

|  |
| --- |
| **# ./executable** |

Normalement, vous devriez voir apparaître le mot «**helloword**» dans votre terminal.

Modifiez votre programme pour qu'il affiche « **Bonjour tout le monde** » avec un retour à ligne à la fin.

## Exercices

* Un nom est saisi au clavier (ex : robert) puis l’ordinateur affiche " Bonjour robert ", (utiliser **scanf** de la librairie **stdlib.h** et **printf**)
* Après avoir entré la longueur et la largeur le programme retourne le périmètre,   
  (utiliser une fonction : **float perimetre(float l,float L) ;**)
* Recherchez la racine réelle d'une équation du premier degré.
* Recherchez les racines réelles d'une équation du second degré.   
  (utilisez la librairie **Math.h** et l’option **–lm** pour la compilation)

**# gcc degre2.c –o degre2 –lm** (-l : lier avec une libraire, m : raccourci pour math)

* Recherchez un nombre aléatoire entre 0 et 999, à chaque essai le programme indique « trop grand » ou « trop petit » et en cas de réussite le nombre d’essais, (utiliser **if**, **do while**, **rand**)

## Elements de correction

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* nom : bonjour.c  \* auteur : Marc Silanus  \* date : 14/02/2011  \*  \* Formation STI2D-SIN / Programmation en C sous linux  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  int main ()  {  char prenom[20];  printf ("Quel est ton prénom ?\n");  scanf ("%19s",prenom);  printf("Bonjour %s...\n",prenom);  return 0;  } |

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* nom : degre1.c  \* auteur : Marc Silanus  \* date : 14/02/2011  \*  \* Formation STI2D-SIN / Programmation en C sous linux  #include <stdio.h>  #include <stdlib.h>  int main ()  {  float a,b,c,x;  a=0;  printf ("Résolution d'une équation du premier degré : ax+b=c\n");  printf ("Entrez la valeur de a (différent de 0) : ");  while(a==0) scanf ("%f",&a);  printf ("Entrez la valeur de b : ");  scanf ("%f",&b);  printf ("Entrez la valeur de c : ");  scanf ("%f",&c);  x=(c-b)/a;  printf("La racine est : x=%f\n",x);  return 0;  } |

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* nom : degre2.c  \* auteur : Marc Silanus  \* date : 14/02/2011  \*  \* Formation STI2D-SIN / Programmation en C sous linux  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  int main ()  {  float a,b,c,x1,x2,delta;  a=0;  printf ("Résolution d'une équation du second degré dans R : ax²+bx+c=0\n");  printf ("Entrez la valeur de a (différente de 0) : ");  while(a==0) scanf ("%f",&a);  printf ("Entrez la valeur de b : ");  scanf ("%f",&b);  printf ("Entrez la valeur de c : ");  scanf ("%f",&c);  delta=((b\*b)-(4\*a\*c));  if(delta>0)  {  x1=(-b-sqrt(delta))/(2\*a);  x2=(-b+sqrt(delta))/(2\*a);  printf("Les racines sont : \nx1=%f\nx2=%f\n",x1,x2);  }  else if(delta==0)  {  x1=(-b)/(2\*a);  printf("Une racine unique : \nx=%f\n",x1);  }  else  printf("Pas de racines réelles\n");  return 0;  } |

|  |
| --- |
| **# gcc degre2.c –o degre2 –lm** (-l : lier avec une libraire, m : raccourci pour math) |

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* nom : perimetre.c  \* auteur : Marc Silanus  \* date : 14/02/2011  \*  \* Formation STI2D-SIN / Programmation en C sous linux  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  float L,l;  float perimetre(float a, float b);  int main(void)  {  printf("Calcul du périmètre d'un rectangle\n");  printf("\nSaisissez la longueur : ");  scanf("%f",&L);  printf("\nSaisissez la largeur : ");  scanf("%f",&l);  printf("\nCalcul du périmètre : %5.2f\n",perimetre(L,l));    }  float perimetre(float a, float b)  {  return (a+b)\*2;  } |

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* nom : chiffre-secret.c  \* auteur : Marc Silanus  \* date : 14/02/2011  \*  \* Formation STI2D-SIN / Programmation en C sous linux  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  int main ()  {  int iSecret, iTente, Compt;  /\* initialisation de la fonction rand \*/  srand ( time(NULL) );  /\* génération du nombre secret \*/  iSecret = rand() % 999;  /\* initialisation du compteur de coups \*/  Compt=0;  do  {  Compt++;  printf ("Entrez un nombre (0 to 999): ");  scanf ("%d",&iTente);  if (iSecret<iTente) printf("Le nombre secret est plus petit !\n");  else if (iSecret>iTente) printf("Le nombre secret est plus grand !\n");  } while (iSecret!=iTente);  printf("Félicitation !\nVous avez trouvé en %d coups\n",Compt);  return 0;  } |