Révisions

Valeur d'accueil et de reconversion en informatique (VARI1)

Daniel Porumbel (dp.cnam@gmail.com)

http://cedric.cnam.fr/~porumbed/vari1/

Plan

Machine informatique : couches génériques

Couche logicielle

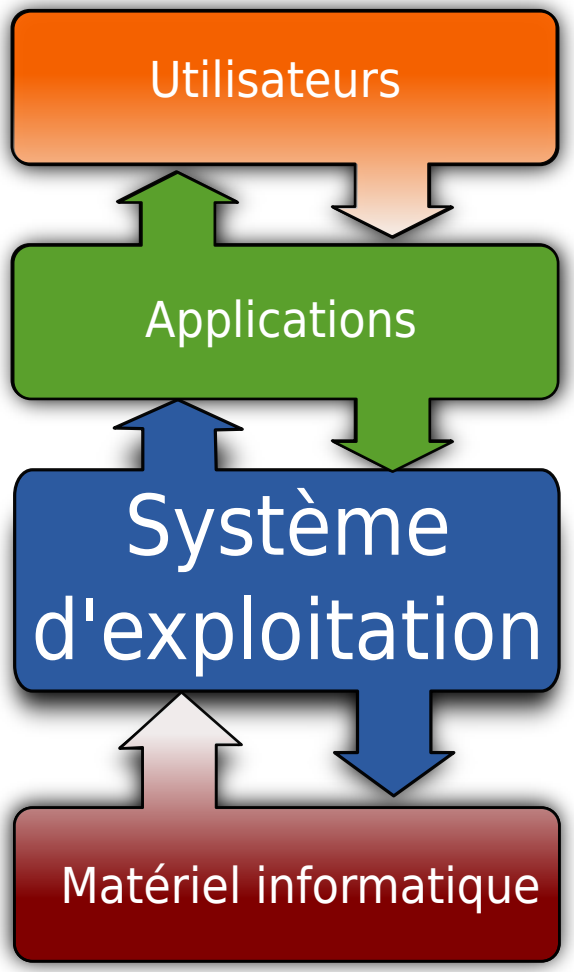
des programmes qui permettent à l'utilisateur de réaliser des taches, y compris `processing` et `java`

Couche Système d'Exploitation

dirige l'utilisation des ressources de la machine par les programmes de la couche logicielle

Couche Matérielle

la machine physique y compris le processeur (CPU), la mémoire vive (RAM), disques durs, clés USB, imprimantes, etc.



Fonctions du système d'exploitation (OS)

Le système d'exploitation : aussi appelé OS, de l'anglais Operating System.

OS = Interface entre le matériel et les logiciels

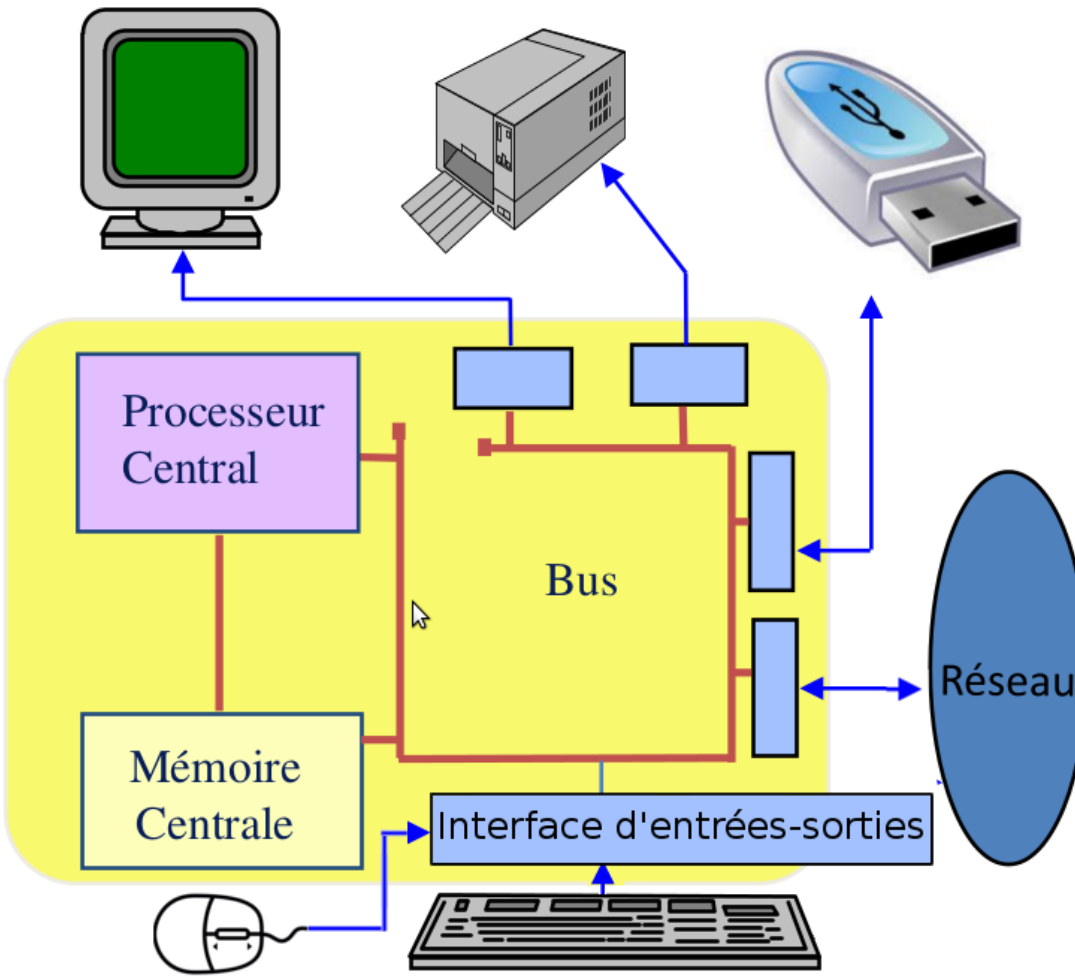
Ressources matérielles : Processeur, mémoire vive (RAM), fichiers, réseaux, interface graphique utilisateur, périphériques, disques durs, contrôle d'accès pour plusieurs usagers simultanément, etc.

Principaux OS

- Linux/Unix ses distributions (Ubuntu, Suze, Debian)
- Windows 95, Windows Vista, Windows 7, etc.
- MacOS, Android (basés sur des noyaux Linux)

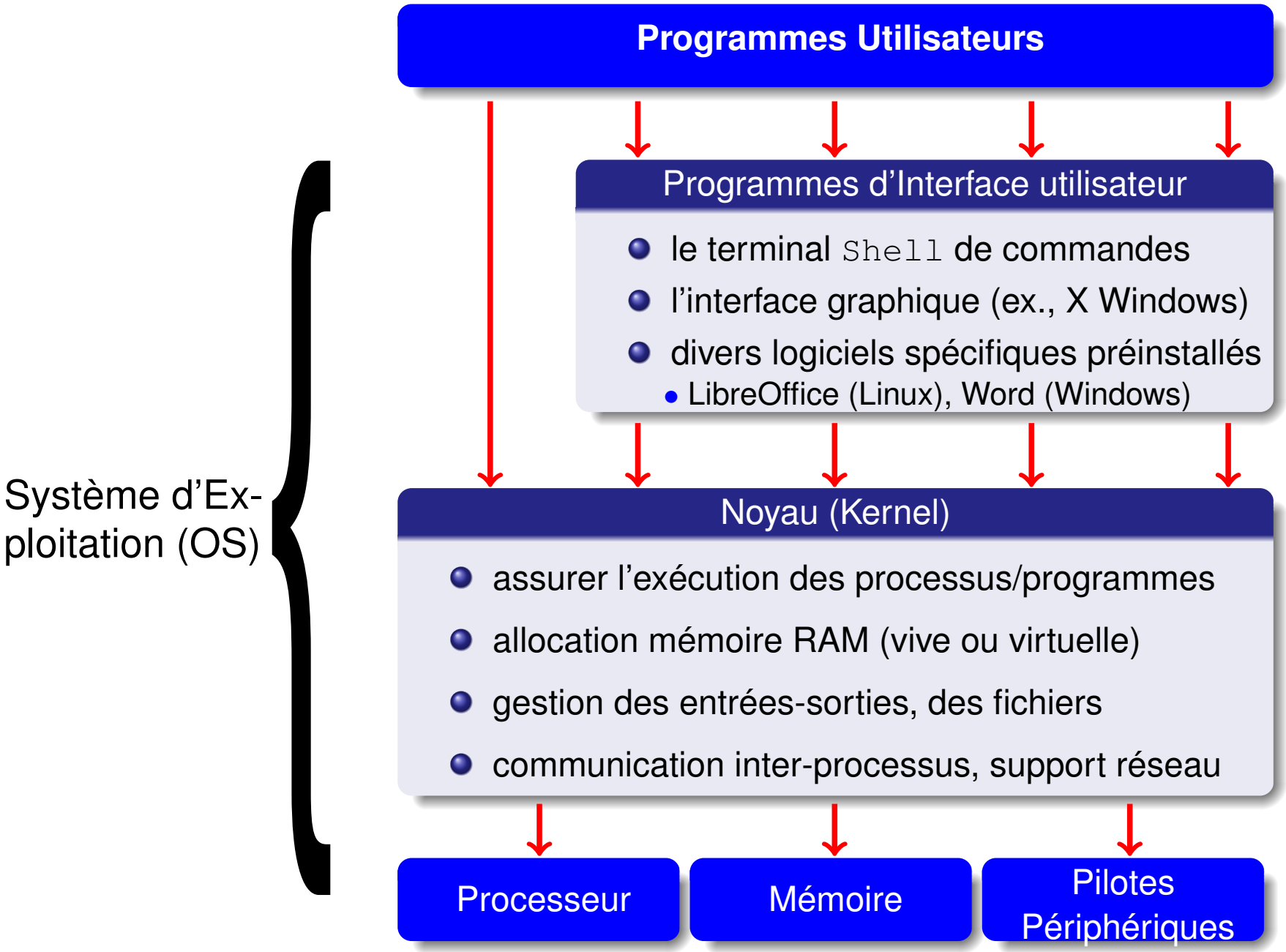
Composants couche physique (hardware)

- le processeur exécute les instructions machine, c'est le cerveau du système
- les mémoires vives (RAM, cache) stockent les données et les instructions
- le bus permet le transfert de données entre les différents composants
  - souvent implémenté sur la carte mère.



- les périphériques : disques durs, clés USB, imprimantes, moniteur (écran), clavier, souris, cartes d'extension (graphique), manettes de jeu, lecteurs de CD/DVD, etc.

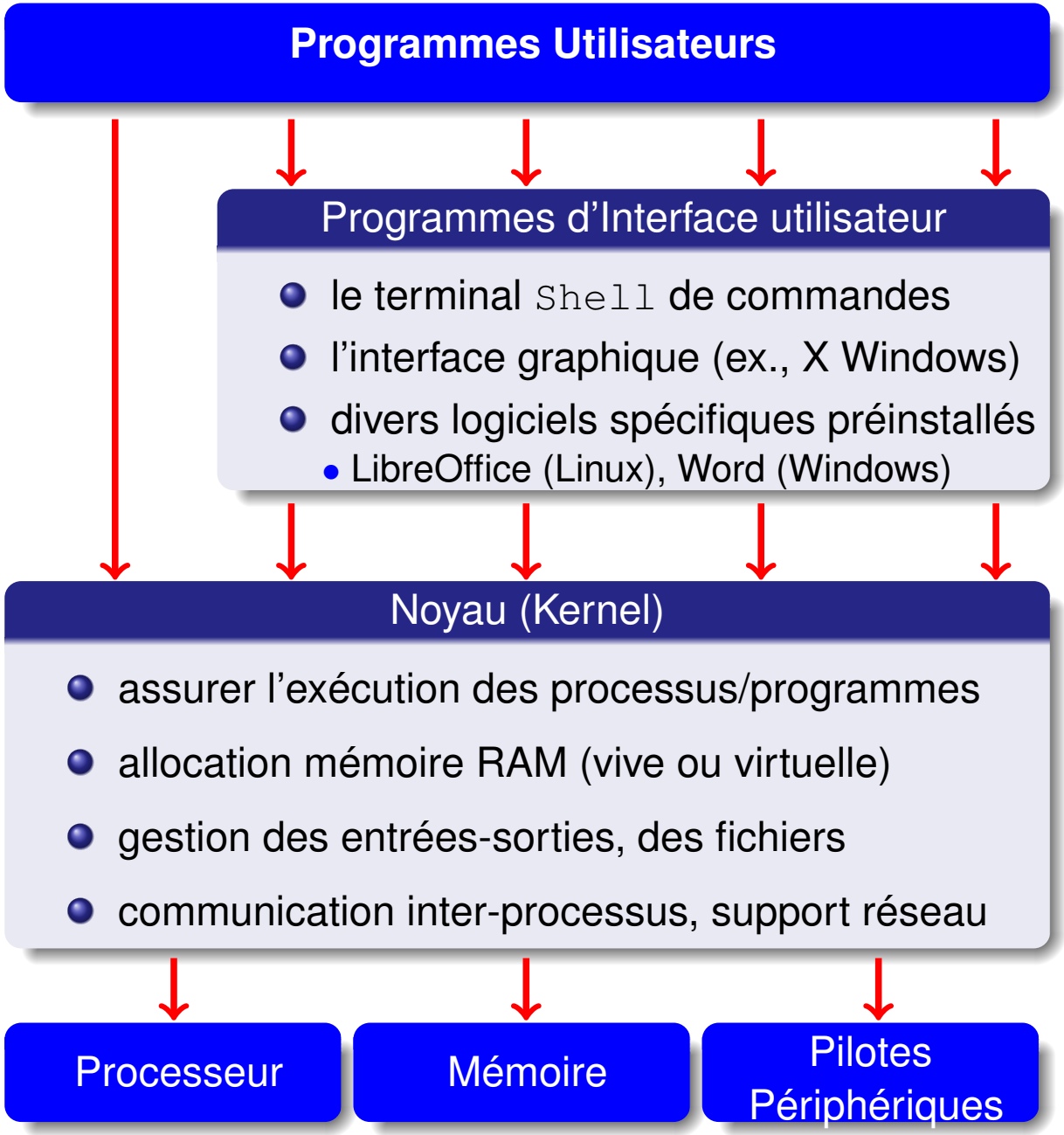
Composition Système d'Exploitation (OS)



Composition Système d'Exploitation (OS)

Pour lancer un programme (ex, `Processing`, navigateur Web, ...), il est d'abord chargé en mémoire et l'OS exécute les instructions

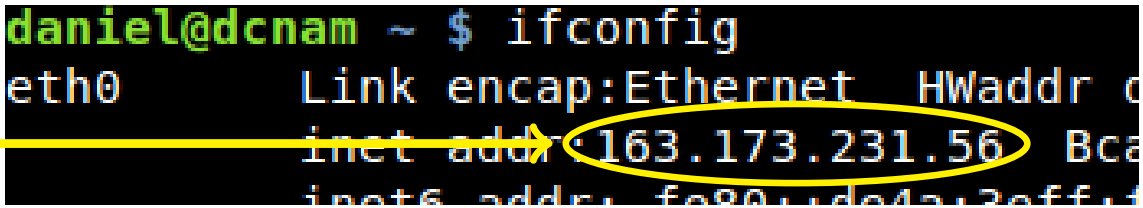
- Appels système aux fonctions de l'interface utilisateur, ex. la fonction `ellipse(...)` est envoyée à un serveur d'affichage
- Appels système au noyau (le premier programme chargé par l'OS pour gérer les fonctionnalités de base)



Les adresses IP

Toute machine connectée possède une adresse IP

- exemples : 10.0.0.1, 163.173.0.1, ou 202.2.1.15
- C'est comme une plaque d'immatriculation,
  - sauf qu'on peut avoir plusieurs IPs si on a plusieurs cartes réseaux
- `ifconfig` : afficher notre adresse IP

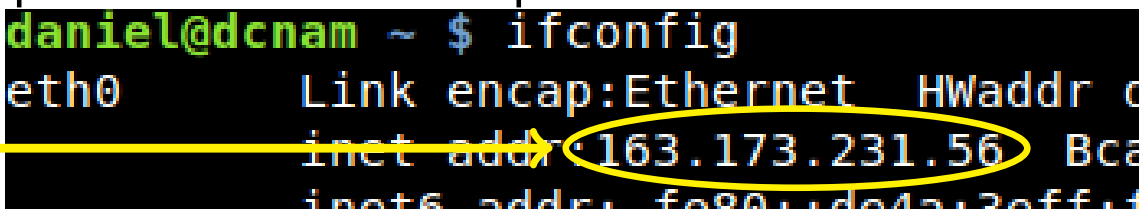


- `route -n` ⟹ afficher l'IP de la passerelle par défaut
  - ex., une box ADSL qui nous connecte à l'Internet
- `ping 163.173.228.2` ⟹ tester si la machine d'adresse IP `163.173.228.2` est active
- `traceroute google.fr` ⟹ voir les machines/routeurs qui nous relient à la machine `google.fr` (en Californie)
  - Un message est transféré de routeur en routeur (une dizaine de routeurs sont traversés) pour l'acheminer à `google.fr`

Les adresses IP

Toute machine connectée possède une adresse IP

- exemples : 10.0.0.1, 163.173.0.1, ou 202.2.1.15
- C'est comme une plaque d'immatriculation,
  - sauf qu'on peut avoir plusieurs IPs si on a plusieurs cartes réseaux
- `ifconfig` : afficher notre adresse IP



- `route -n` $\Longrightarrow$ afficher l'IP de la passerelle par défaut
  - ex., une box ADSL qui nous connecte à l'Internet
- `ping 163.173.228.2` $\Longrightarrow$ tester si la machine d'adresse IP `163.173.228.2` est active
- `traceroute google.fr` $\Longrightarrow$ voir les machines/routeurs qui nous relient à la machine `google.fr`[1] (en Californie)
  - Un message est transféré de routeur en routeur (une dizaine de routeurs sont traversés) pour l'acheminer à `google.fr`

1. Utiliser `www.iplocation.net` pour voir la location géographique des routeurs.

Commandes Réseau Linux

La commande `ifconfig` affiche ou modifie l'IP

- `sudo ifconfig eth0 10.0.0.1/8` $\Longrightarrow$ nouvelle IP [2]
  - Si le réseau utilise un service/serveur DHCP (Dynamic Host Configuration Protocol), l'IP est configuré automatiquement (ou avec `dhclient`), et non pas avec `ifconfig`
- En plus d'une adresse IP, la machine possède une adresse de diffusion (en. : *broadcast*) qui désigne toutes les machines du (sous-)réseau

La commande `ssh SERVEUR` permet de se connecter à la machine `SERVEUR` en mode console

- L'option `-X` de `ssh` permet d'afficher les commandes graphiques sur le serveur d'affichage (X) de la machine locale

2. Il faut parfois taper "`stop network-manager`" pour arrêter le gestionnaire graphique de réseaux, sinon il peut annuler l'effet d'`ifconfig`.

Commandes Réseau Linux

La commande `ifconfig` affiche ou modifie l'IP

- `sudo ifconfig eth0 10.0.0.1/8` $\Longrightarrow$ nouvelle IP [2]
  - Si le réseau utilise un service/serveur DHCP (Dynamic Host Configuration Protocol), l'IP est configuré automatiquement (ou avec `dhclient`), et non pas avec `ifconfig`
- En plus d'une adresse IP, la machine possède une adresse de diffusion (en. : *broadcast*) qui désigne toutes les machines du (sous-)réseau

La commande `ssh SERVEUR` permet de se connecter à la machine `SERVEUR` en mode console

- L'option $-X$ de `ssh` permet d'afficher les commandes graphiques sur le serveur d'affichage (X) de la machine locale

2. Il faut parfois taper "`stop network-manager`" pour arrêter le gestionnaire graphique de réseaux, sinon il peut annuler l'effet d'`ifconfig`.

Exemple d'utilisation couche Applications

Pour aller sur `www.cnam.fr` :

1. Le navigateur web demande au service DNS (Domain Name Server) l'adresse IP du serveur `www.cnam.fr`
   - C'est comme si on tapait `dig www.cnam.fr`
2. Le DNS retourne `163.173.128.40` par exemple.
3. Le navigateur envoie une requête HTTP à cette adresse IP
4. Le serveur web répond avec un fichier `html` (⊕ des scripts `javascript` éventuellement)

- La communication passe par une connexion TCP sur le port 80 de `www.cnam.fr` (port supérieur à 1024 pour la machine locale)

Le site peut demander de déposer un cookie qui stocke une valeur dans le navigateur. Cette valeur peut être demandée plus tard au navigateur, ex, pour faire du pistage (publicitaire).

Exemple d'utilisation couche Applications

Pour aller sur `www.cnam.fr` :

1. Le navigateur web demande au service DNS (Domain Name Server) l'adresse IP du serveur `www.cnam.fr`
   - C'est comme si on tapait `dig www.cnam.fr`
2. Le DNS retourne `163.173.128.40` par exemple.
3. Le navigateur envoie une requête HTTP à cette adresse IP
4. Le serveur web répond avec un fichier `html` (⊕ des scripts `javascript` éventuellement)
   - La communication passe par une connexion TCP sur le port 80 de `www.cnam.fr` (port supérieur à 1024 pour la machine locale)

Le site peut demander de déposer un cookie qui stocke une valeur dans le navigateur. Cette valeur peut être demandée plus tard au navigateur, ex, pour faire du pistage (publicitaire).

Exemple d'utilisation couche Applications

Pour aller sur `www.cnam.fr` :

1. Le navigateur web demande au service DNS (Domain Name Server) l'adresse IP du serveur `www.cnam.fr`
   - C'est comme si on tapait `dig www.cnam.fr`
2. Le DNS retourne `163.173.128.40` par exemple.
3. Le navigateur envoie une requête HTTP à cette adresse IP
4. Le serveur web répond avec un fichier `html` (⊕ des scripts `javascript` éventuellement)
   - La communication passe par une connexion TCP sur le port 80 de `www.cnam.fr` (port supérieur à 1024 pour la machine locale)

Le site peut demander de déposer un cookie qui stocke une valeur dans le navigateur. Cette valeur peut être demandée plus tard au navigateur, ex, pour faire du pistage (publicitaire).

Exemple d'utilisation couche Applications

Pour aller sur `www.cnam.fr` :

1. Le navigateur web demande au service DNS (Domain Name Server) l'adresse IP du serveur `www.cnam.fr`
   - C'est comme si on tapait `dig www.cnam.fr`
2. Le DNS retourne `163.173.128.40` par exemple.
3. Le navigateur envoie une requête HTTP à cette adresse IP
4. Le serveur web répond avec un fichier `html` ($\oplus$ des scripts `javascript` éventuellement)
   - La communication passe par une connexion TCP sur le port 80 de `www.cnam.fr` (port supérieur à 1024 pour la machine locale)

Le site peut demander de déposer un cookie qui stocke une valeur dans le navigateur. Cette valeur peut être demandée plus tard au navigateur, ex, pour faire du pistage (publicitaire).

Exemple d'utilisation couche Applications

Pour aller sur `www.cnam.fr` :

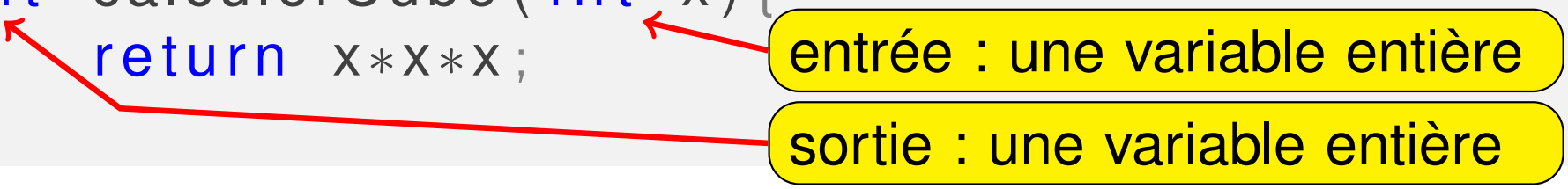
1. Le navigateur web demande au service DNS (Domain Name Server) l'adresse IP du serveur `www.cnam.fr`
   - C'est comme si on tapait `dig www.cnam.fr`
2. Le DNS retourne `163.173.128.40` par exemple.
3. Le navigateur envoie une requête HTTP à cette adresse IP
4. Le serveur web répond avec un fichier `html` ($\oplus$ des scripts `javascript` éventuellement)
   - La communication passe par une connexion TCP sur le port 80 de `www.cnam.fr` (port supérieur à 1024 pour la machine locale)

Le site peut demander de déposer un cookie qui stocke une valeur dans le navigateur. Cette valeur peut être demandée plus tard au navigateur, ex, pour faire du pistage (publicitaire).

Plan

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```



Exemple de définition de méthode :

```
void direBonjour(String nomPersonne){
    println("Bonjour␣"+nomPersonne);
}
```

Exemples d'appels de fonctions/méthodes

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```

entrée : une variable entière

sortie : une variable entière

Exemple de définition de méthode :

```
void direBonjour(String nomPersonne){
    println("Bonjour "+nomPersonne);
}
```

Exemples d'appels de fonctions/méthodes

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```

entrée : une variable entière

sortie : une variable entière

Exemple de définition de méthode :

```
void direBonjour(String nomPersonne){
    println("Bonjour "+nomPersonne);
}
```

Exemples d'appels de fonctions/méthodes

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Variables : globales, locales ou arguments

Comment corriger et lancer ce programme ?

```
int x;                              //x=variable globale
int detMin(int a, int b){           //a et b: arguments
    println(x);                     //x visible partout
    if(a<=b)
        return a;
    return b;                       //on arrive ici si b<a
    println("Toto???");             //on arrive jamais ici
}
void setup(){
    x = 8;
    int y = 9;                      //y: variable locale
    y = detMin(x,y);
    println(y);
    println(a);                     //erreur: a n'est pas
}                                   //déclaré ici!
```

Variables : globales, locales ou arguments

Comment corriger et lancer ce programme ?

```
int x;                              //x=variable globale
int detMin(int a, int b){           //a et b: arguments
    println(x);                     //x visible partout
    if(a<=b)
        return a;
    return b;                       //on arrive ici si b<a
    println("Toto???");             //on arrive jamais ici
}
void setup(){
    x = 8;
    int y = 9;                      //y: variable locale
    y = detMin(x,y);
    println(y);
    println(a);                     //erreur: a n'est pas
}
```

unreachable code

a can not be resolved to a variable

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom, char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3;//conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T');//=> "Salut T Toto"
direSalut("Titi", 'S');//=> "Salut S Titi"
float m = calculerMoyenne(7,7,8);//m=7.33
...
```

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom,char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3;//conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T');//=> "Salut T Toto"
direSalut("Titi", 'S');//=> "Salut S Titi"
float m = calculerMoyenne(7,7,8);//m=7.33
...
```

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom,char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3;//conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T');//=> "Salut T Toto"
direSalut("Titi", 'S');//=> "Salut S Titi"
float m = calculerMoyenne(7,7,8);//m=7.33
...
```

Le passage des paramètres aux fonctions

```
int f(int x, int y){
    return x+y;
}
void setup(){
    int somme;
    somme = f(2,3);      //somme devient 5
    int a=4, b=5;
    somme = f(a,b);      //somme devient 9
    println(somme);
}
```

- x et y sont visibles/connus uniquement à l'intérieur (du corps) de la fonction f(...)
- a et b sont visibles uniquement à l'intérieur (du corps) de la fonction setup()

Le passage des paramètres aux fonctions

```
int f(int x, int y){
    return x+y;
}
void setup(){
    int somme;
    somme = f(2,3);     //somme devient 5
    int a=4, b=5;
    somme = f(a,b);     //somme devient 9
    println(somme);
}
```

les valeurs de a et b sont affectées à x et y

- x et y sont visibles/connus uniquement à l'intérieur (du corps) de la fonction f(...)
- a et b sont visibles uniquement à l'intérieur (du corps) de la fonction setup()

Plan

Introduction boucles

Soit le programme à gauche

- observer les instructions if répétitives
- comment automatiser ces instructions ?

Sans boucle

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  if(tab[1]<min)
      min = tab[1];
  if(tab[2]<min)
      min = tab[2];
  if(tab[3]<min)
      min = tab[3];
  println(min);
}
```

Introduction boucles

Soit le programme à gauche

- observer les instructions if répétitives
- comment automatiser ces instructions ? utiliser la boucle for

Sans boucle

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  if(tab[1]<min)
      min = tab[1];}i= 1
  if(tab[2]<min)
      min = tab[2];}i= 2
  if(tab[3]<min)
      min = tab[3];}i= 3
  println(min);
}
```

Introduction boucles

Soit le programme à gauche

- observer les instructions if répétitives
- comment automatiser ces instructions ? utiliser la boucle for

Sans boucle

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  if(tab[1]<min)
      min = tab[1];      } i= 1
  if(tab[2]<min)
      min = tab[2];      } i= 2
  if(tab[3]<min)
      min = tab[3];      } i= 3
  println(min);
}
```

Boucle For

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  int i;
  for(i=1;i<4;i++){
    if(tab[i]<min)
        min = tab[i];
  }
  println(min);
}
```

Introduction boucles

Soit le programme à gauche

- observer les instructions if répétitives
- comment automatiser ces instructions ?

Sans boucle

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  if(tab[1]<min)
      min = tab[1];      } i= 1
  if(tab[2]<min)
      min = tab[2];      } i= 2
  if(tab[3]<min)
      min = tab[3];      } i= 3
  println(min);
}
```

Boucle For

```
void setup(){
  int[] tab = new int[4];
  tab[0]=5;tab[1]=4;
  tab[2]=2;tab[3]=1;
  int min=tab[0];
  int i;
  for(i=1;i<4;i++){
    if(tab[i]<min)
      min = tab[i];
  }
  println(min);
}
```

condition pour continuer

initialisation *i*

Incrémentation avant l'itération suivante

Avantages boucles : le passage à l'échelle, trouver le minimum d'un tableau de 1000 valeurs

```
void setup(){
  int[] tab = new int[1000];
  int i;
  //initialisation tableau
  for(i=0;i<1000;i++) //une seule instruction=>
    tab[i]= i;        //pas besoin d'accolades
  //calcul minimum
  int min=tab[0];
  for(i=1;i<1000;i++){
    if(tab[i]<min)
        min = tab[i];
  }
  println(min);
}
```

Une fonction avec une boucle

```
float calcMin(float[] t){
  float min=t[0];
  for(int i=1;i<1000;i++){//i déclaré dans le
    if(t[i]<min)              //1er champs du for
      min = t[i];
  }
  return min;
}
void setup(){
  float[] tab = new float[1000];
  int i;
  for(i=0;i<1000;i++)         //tableau avec des
    tab[i]= random(999999);//vals aléatoires
  float m = calcMin(tab);
  println(m);
}
```

Quel est le résultat des deux programmes ?

```
println("*****");
println("*****");
println("*****");
println("*****");
println("*****");
```

```
int i;
for(i=1;i<=5;i++)
    println("*****");
```

Quel est le résultat du code ?

```
void setup(){
    size(400,400);
    int i;
    for(i=0;i<15;i++){
        // fill(10*i,0,0);
        fill(i*25,0,0);
        rect(i*10,i*20,10*i, 10*i);
    }
}
```

Boucles `for` et boucles `while`

- La plus classique boucle `for`

```
for(int i=0;i<100;i++){
    ...//faire quelque chose (100 fois)
}
```

- La plus classique boucle `while` (tant que)

```
int i = 0;
while(i<100){
    ...//faire quelque chose
    ...//100 fois si la seule ligne qui
    ...//modifie i est la suivante
    i = i+1;
}
```