

# Java sécurité

# Quelques éléments de Sécurité bas-niveau

- Au niveau du code
  - les OS assurent certaines protections pour la sécurité bas niveau
  - anti-virus
  - pare-feux
  - sandbox Java



# Sécurité bas-niveau





















- Sécurité bas-niveau:
  - « bugs » des programmes
  - éviter les bugs par des bonnes pratiques de conceptions et d'implémentations
  - Exemple: un « warning » au cours d'une compilation correspond à une faille potentielle de sécurité

# Buffer overflow

- Buffer overflow (accès à une zone non-allouée)
- bug (du programme) classique en C C++provoque un crash du programme mais permet à un attaquant de
  - voler des informations privées
  - corrompre des données
  - exécuter un code malveillant

# Buffer overflow

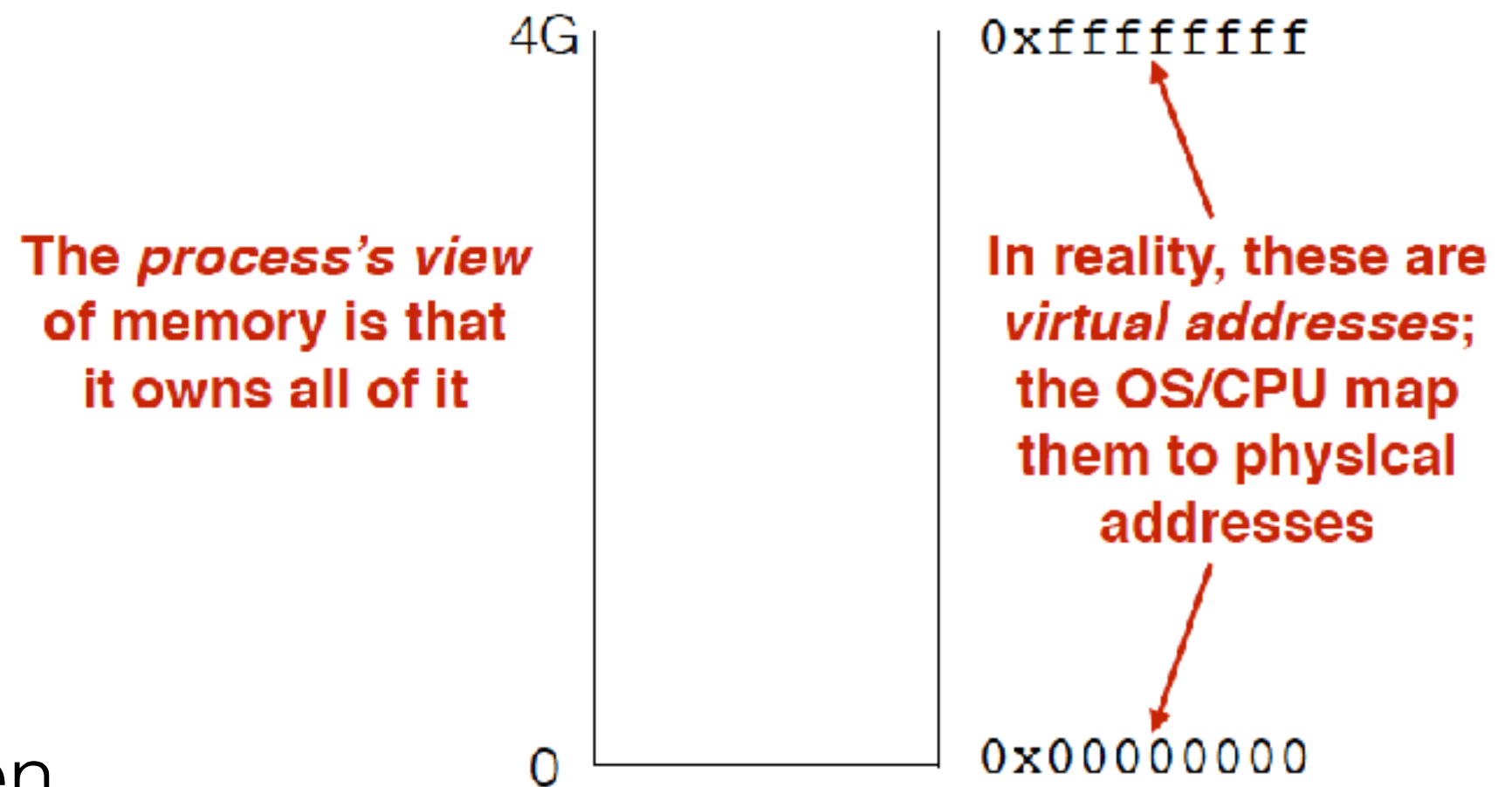
- un classique en C , C++
- pour mémoire:

Language Rank	Types	Spectrum Ranking
1. Java	  	100.0
2. C	  	99.2
3. C++	  	95.5
4. Python	 	93.4
5. C#	  	92.2
6. PHP		84.6
7. Javascript	 	84.3
8. Ruby		78.6
9. R		74.0
10. MATLAB		72.6

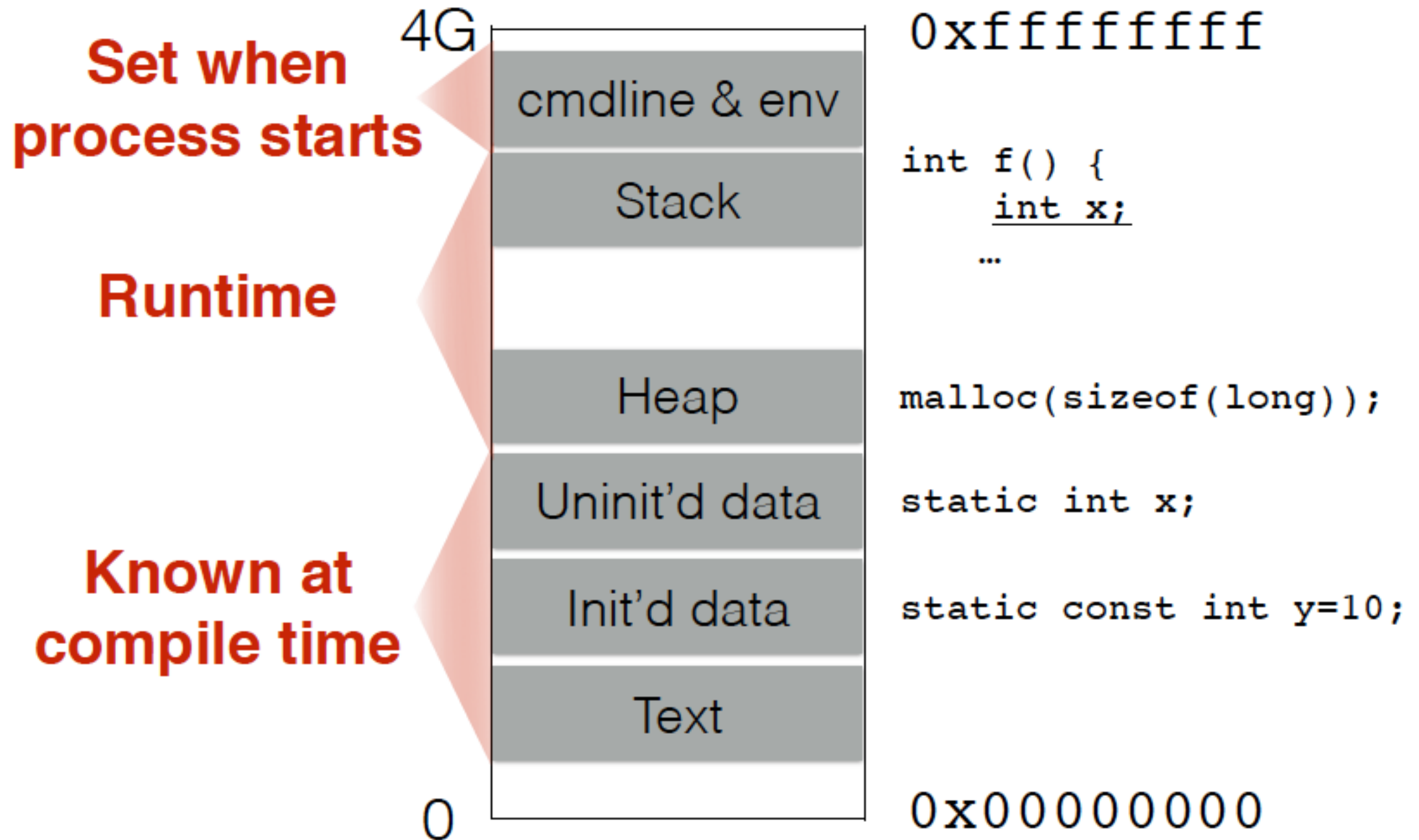
<http://spectrum.ieee.org/static/interactive-the-top-programming-languages>

# pour mémoire

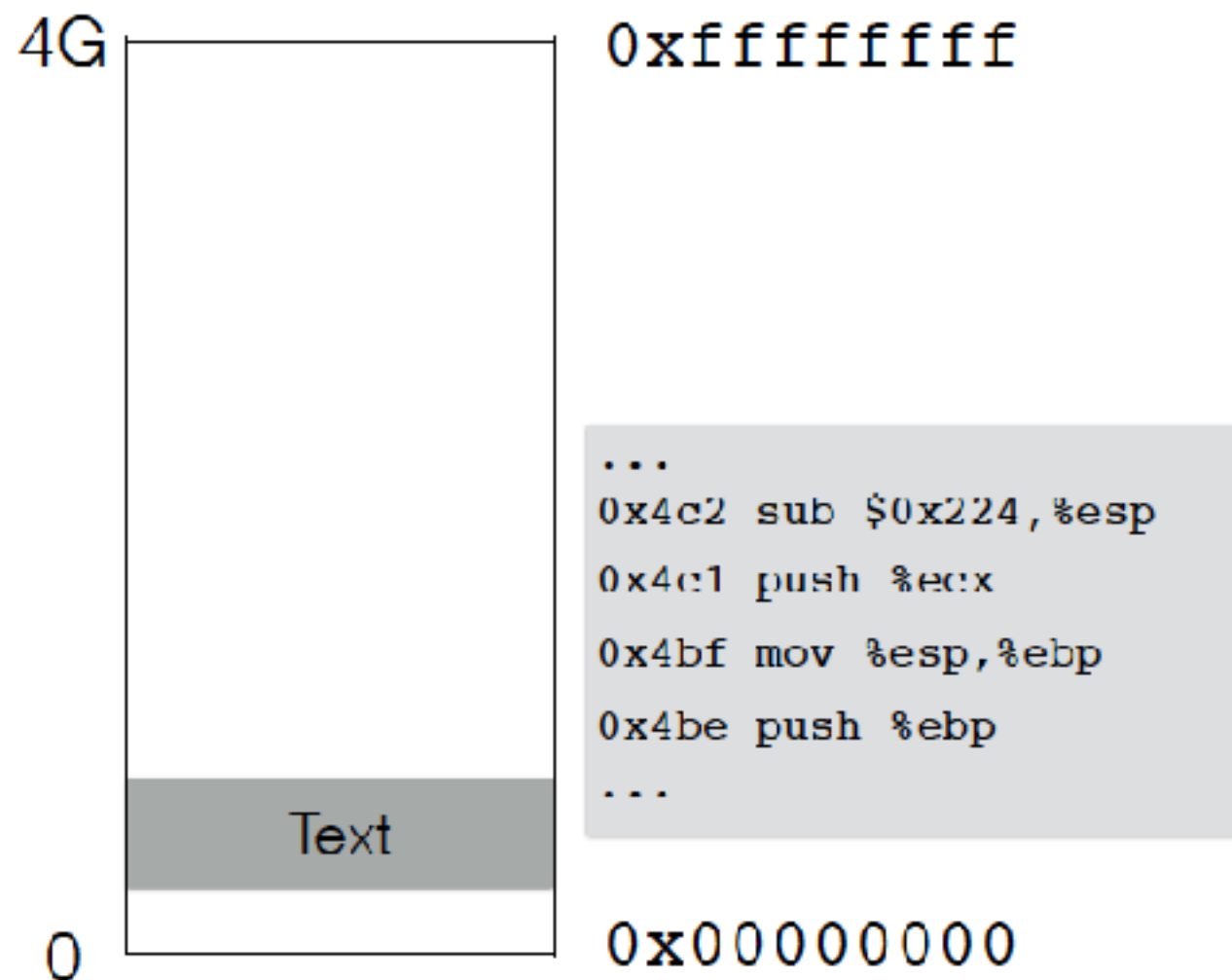
un programme en  
mémoire



# Pour mémoire...



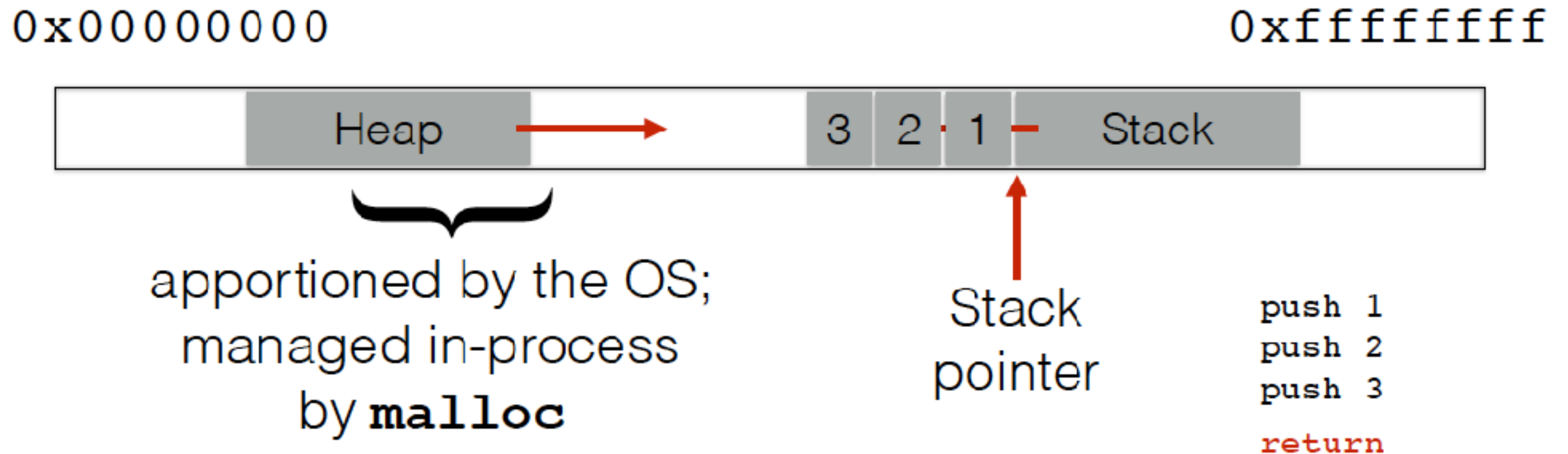
# Pour mémoire



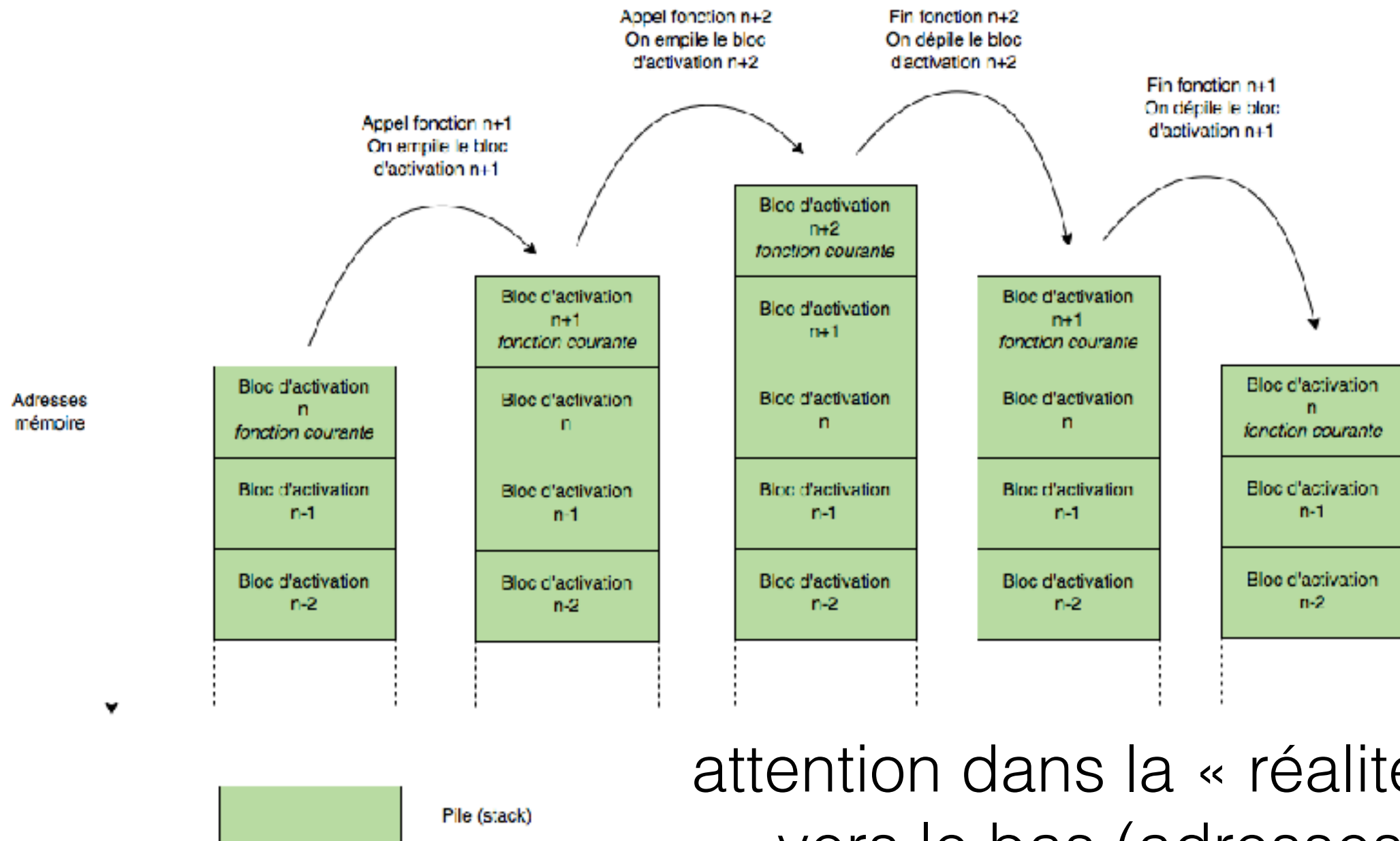
le code est dans la  
zone de texte



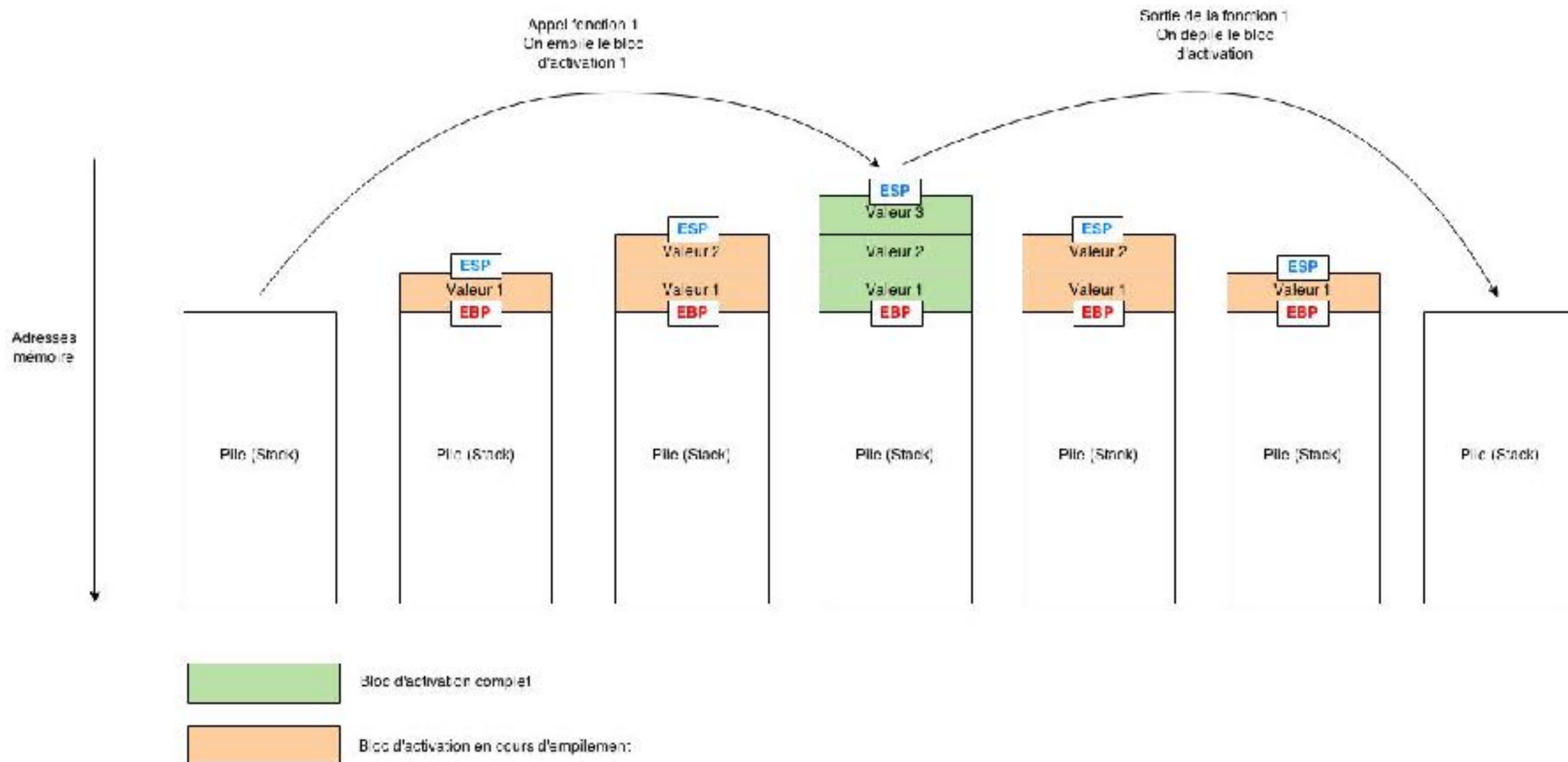
# Allocation



# Pile...



attention dans la « réalité » la pile va vers le bas (adresses basses)

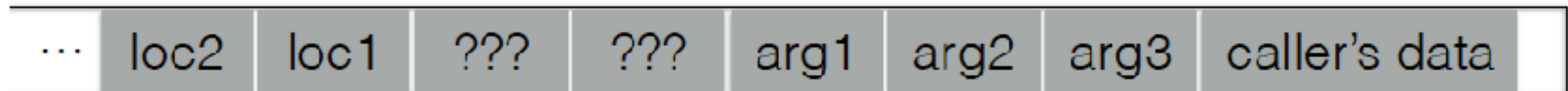


%ebp est l'adresse du début de la stack frame  
 %esp est l'adresse du haut de la pile

# allocation sur la pile

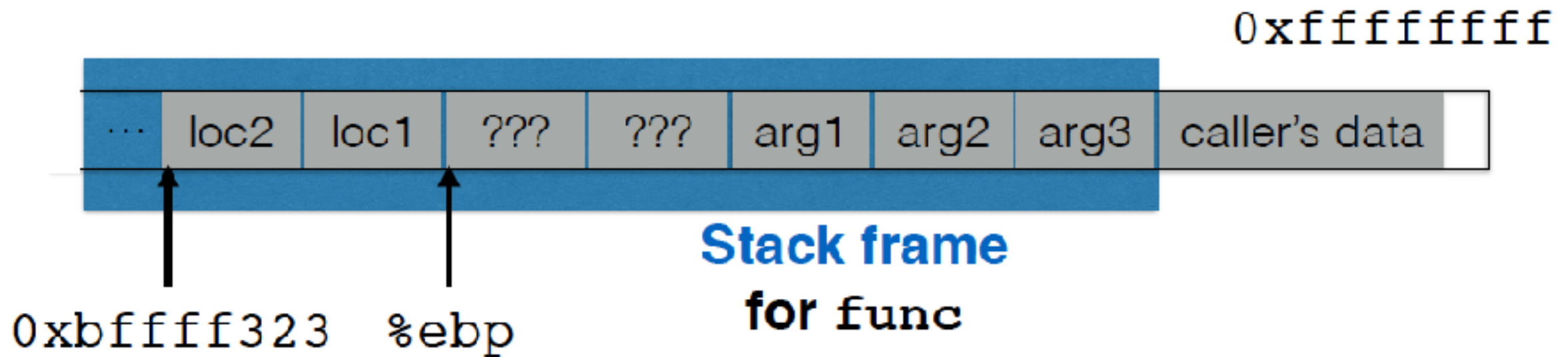
```
void func(char *arg1, int arg2, int arg3)
{
    char loc1[4]
    int  loc2;
    ...
}
```

0xffffffff



**Local variables  
pushed in the  
same order as  
they appear  
in the code**

**Arguments  
pushed in  
reverse order  
of code**

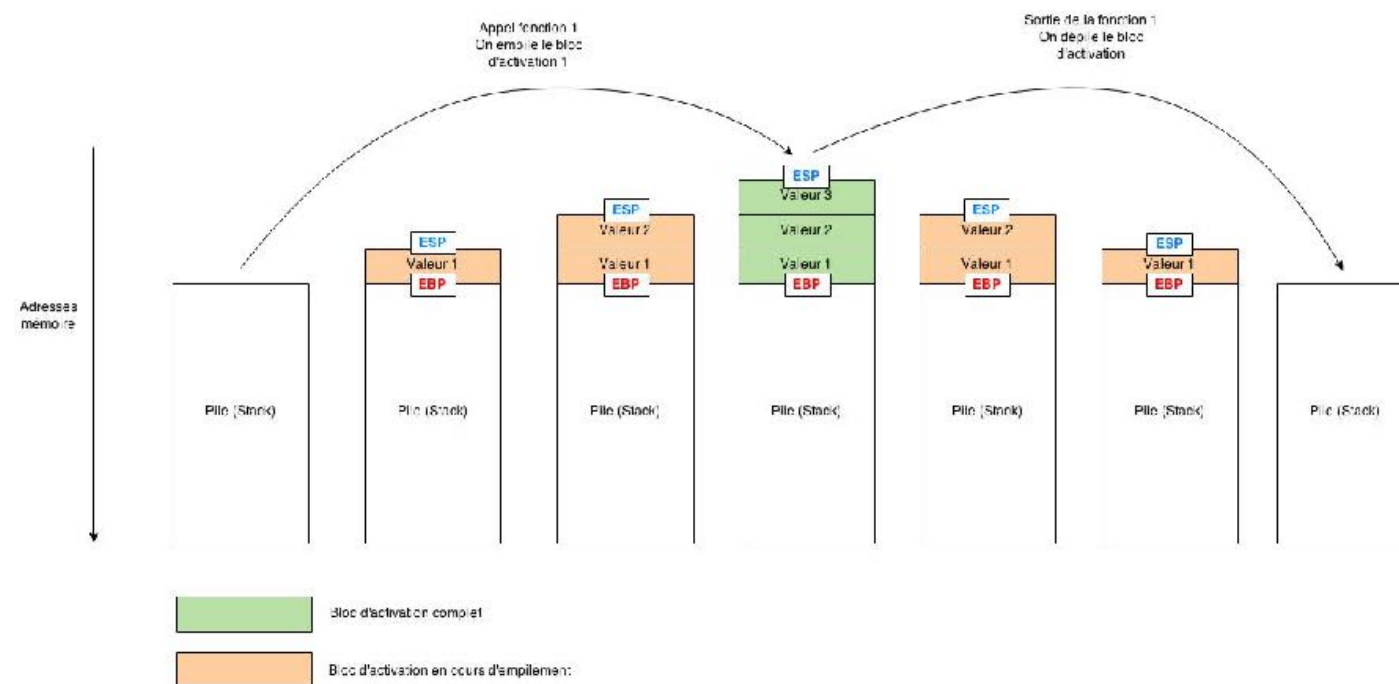


loc2 est à 8 byte avant %ebp

%esp est l'adresse du haut de la pile

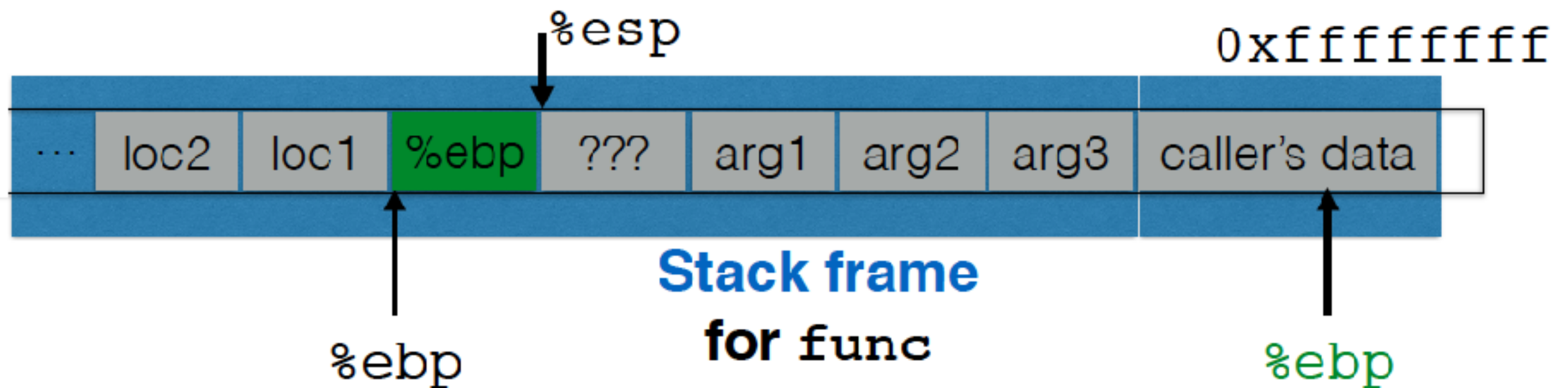
%ebp est l'adresse du début de la stack frame

%eip est l'adresse de l'instruction courante



# Pour le retour

```
int main()  
{  
    ...  
    func("Hey", 10, -3);  
    ...  
}
```



push %ebp  
set %ebp à la valeur courante (%esp)  
set %ebp to (%ebp) au retour

```

#include <stdio.h>
#include <stdlib.h>

int reponse(int a, int b, int c) {
    return a+b+c;
}

int main() {
    int result;
    result = reponse(4, 8, 42);
}

```

Dump of assembler code for function main:

```

0x080483a5 <+0>:    push    ebp
0x080483a6 <+1>:    mov     ebp,esp
0x080483a8 <+3>:    sub     esp,0x1c
0x080483ab <+6>:    mov     DWORD PTR [esp+0x8],0x2a
0x080483b3 <+14>:   mov     DWORD PTR [esp+0x4],0x8
0x080483bb <+22>:   mov     DWORD PTR [esp],0x4
0x080483c2 <+29>:   call    0x8048394
0x080483c7 <+34>:   mov     DWORD PTR [ebp-0x4],eax
0x080483ca <+37>:   leave
0x080483cb <+38>:   ret

```

End of assembler dump.

(gdb) disas reponse

Dump of assembler code for function reponse:

```

0x08048394 <+0>:    push    ebp
0x08048395 <+1>:    mov     ebp,esp
0x08048397 <+3>:    mov     eax,DWORD PTR [ebp+0xc]
0x0804839a <+6>:    mov     edx,DWORD PTR [ebp+0x8]
0x0804839d <+9>:    lea     eax,[edx+eax*1]
0x080483a0 <+12>:   add     eax,DWORD PTR [ebp+0x10]
0x080483a3 <+15>:   pop     ebp
0x080483a4 <+16>:   ret

```

End of assembler dump.



# appel de fonction

- fonction appelante:
  1. push des arguments sur la pile (en ordre inversé)
  2. push l'adresse de retour
  3. aller à l'adresse de la fonction
- fonction appelée
  4. push du pointeur de l'ancienne frame sur la pile (%ebp)
  5. set le pointeur de frame (%ebp) à la valeur de la fin actuelle de pile
  6. push les variables locales sur la pile
- retour de la fonction
  7. reset la précédente stack frame %esp=%ebp, %ebp=(%ebp)
  8. aller à l'adresse de retour: %eip=4(%esp)



# Buffer overflow:

- Buffer: zone contiguë de mémoire associée à une variable. Une String en C est un tableau de char terminé par '\0'
- Overflow: écrire dans le buffer plus que la zone réservée
- ce qui est écrit en trop « écrase » ou modifie une zone non allouée pour ça... un expert peut savoir laquelle et utiliser le buffer overflow de façon malveillante

# Exemple

```
void func(char *arg1)
{
    char buffer[4];
    strcpy(buffer, arg1);
    ...
}

int main()
{
    char *mystr = "AuthMe!";
    func(mystr);
    ...
}
```

sets %ebp to 0x0021654d

M e ! \0					
	A	u	t	h	
	4d	65	21	00	
	%eip		&arg1		

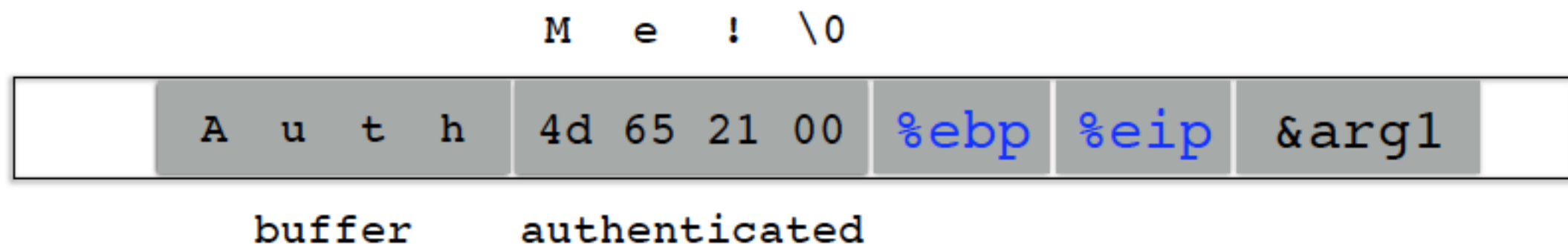
SEGFAULT (0x00216551)

# Exemple

```
void func(char *arg1)
{
    int authenticated = 0;
    char buffer[4];
    strcpy(buffer, arg1);
    if(authenticated) { ...
}

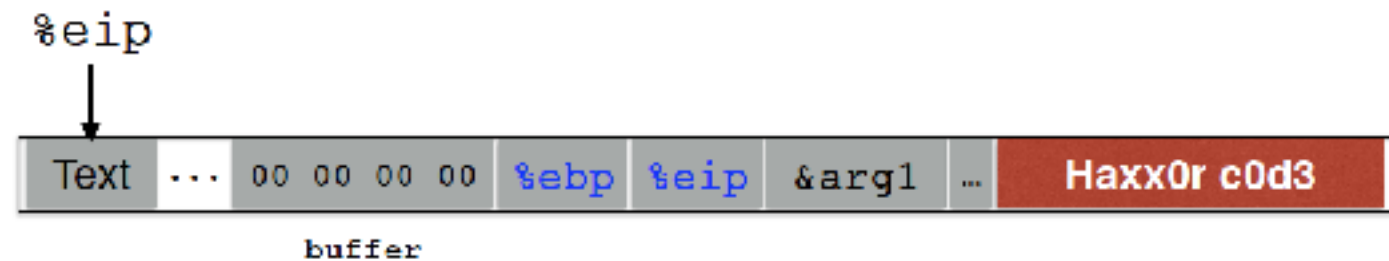
int main()
{
    char *mystr = "AuthMe!";
    func(mystr);
    ...
}
```

strcpy écrit  
jusqu'à '/0':  
on peut même de  
cette façon  
exécuter du code



# injection de code

```
void func(char *arg1)
{
    char buffer[4];
    sprintf(buffer, arg1);
    ...
}
```



- (1) Load my own code into memory
- (2) Somehow get %eip to point to it

# Injection de code...

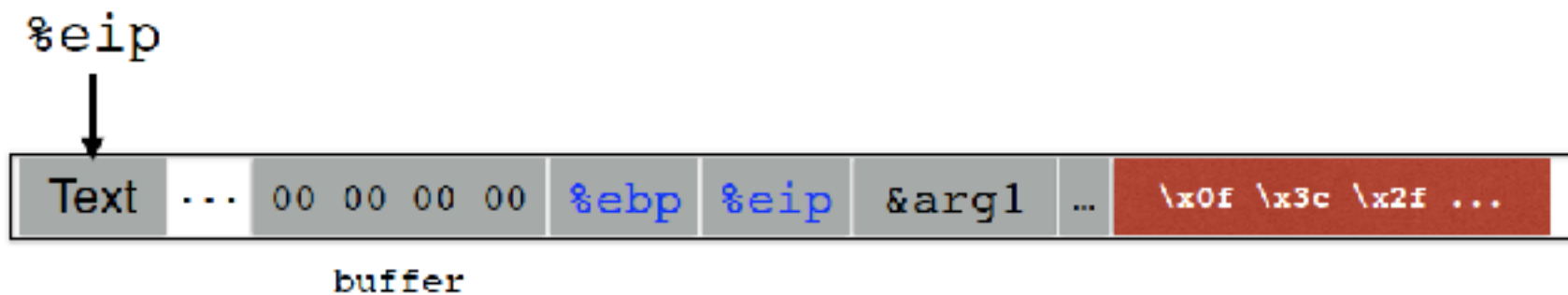
- Mais il faut:
  - du code machine (prêt à être exécuté)
  - sans '\0'
  - sans utiliser le loader
- un code permettant l'accès au system:  
exemple shell

```
#include <stdio.h>
int main( ) {
    char *name[2];
    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL);
}
```

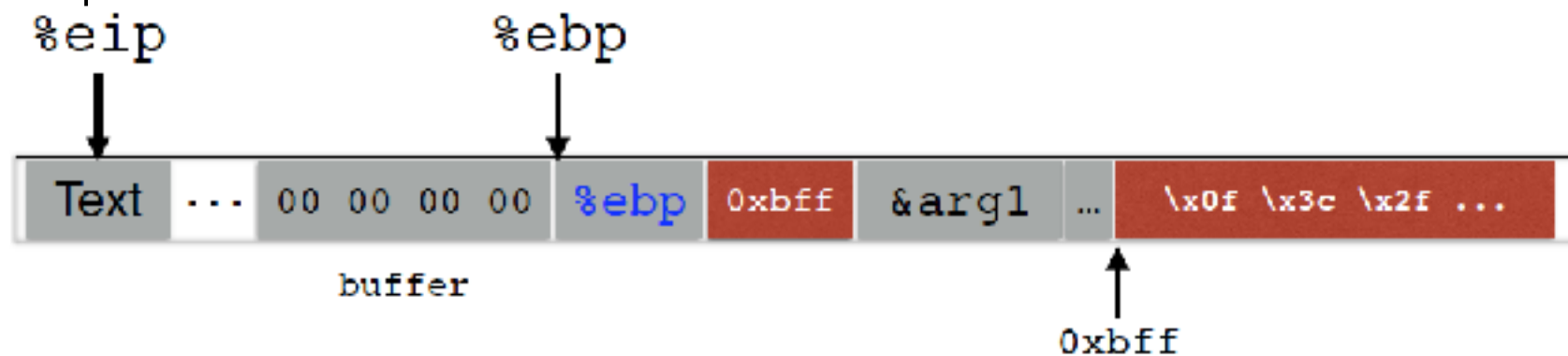
Assembly	xorl %eax, %eax	"\x31\xc0"	Machine code
	pushl %eax	"\x50"	
	pushl \$0x68732f2f	"\x68""//sh"	
	pushl \$0x6e69622f	"\x68""/bin"	
	movl %esp,%ebx	"\x89\xe3"	
	pushl %eax	"\x50"	
	...	...	

# Injection de code

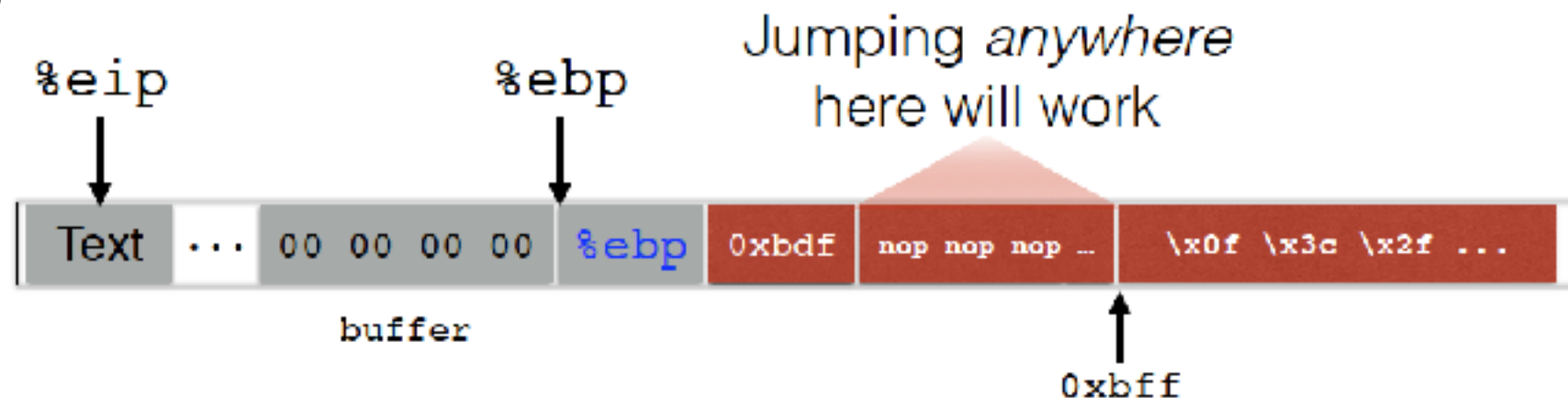
- En plus: faire un « jump » vers le code injecté, (où est-il?)



pirater le `%eip`.



des nop



# mémoire

- Il existe de nombreuses autres attaques...
  - heap overflow: overflow d'un buffer alloué par malloc

```
typedef struct _vulnerable_struct {  
    char buff[MAX_LEN];  
    int (*cmp)(char*,char*);  
} vulnerable;  
  
int foo(vulnerable* s, char* one, char* two)  
{  
    strcpy( s->buff, one );    copy one into buff  
    strcat( s->buff, two );    copy two into buff  
    return s->cmp( s->buff, "file://foobar" );  
}
```

si  $\text{strlen}(\text{one}) + \text{strlen}(\text{two}) > \text{MAX\_LEN}$   
s->cmp est modifié!

# Integer overflow

```
void vulnerable()
{
    char response;
    int nresp = packet_get_int();
    if (nresp > 0) {
        response = malloc(nresp*sizeof(char*));
        for (i = 0; i < nresp; i++)
            response[i] = packet_get_string(NULL);
    }
```

**HUGE** **Wrap-around** **Overflow**

- If we set nresp to 1073741824 and sizeof(char\*) is 4
- then nresp\*sizeof(char\*) overflows to become 0
- subsequent writes to allocated response overflow it



# Dangling pointer

- Pointeur qui ne pointe pas vers le type d'objet approprié:
  - après un free, le pointeur est libéré mais peut continuer à être utilisé par le programme (bug)
  - la zone utilisée peut être modifiée.

# Des solutions?

- stack smashing attacks:
  - interdire d'exécuter du code ne provenant pas de la zone de texte
  - charger les librairies de façon imprévisible
  - ...
- changer (randomisation) les mouvements de la pile
- contrôler les flux: calculer à la compilation le graphe des appels, surveiller à l'exécution...
- Allocateur sûr
- typage sûr
- ...

Surtout:

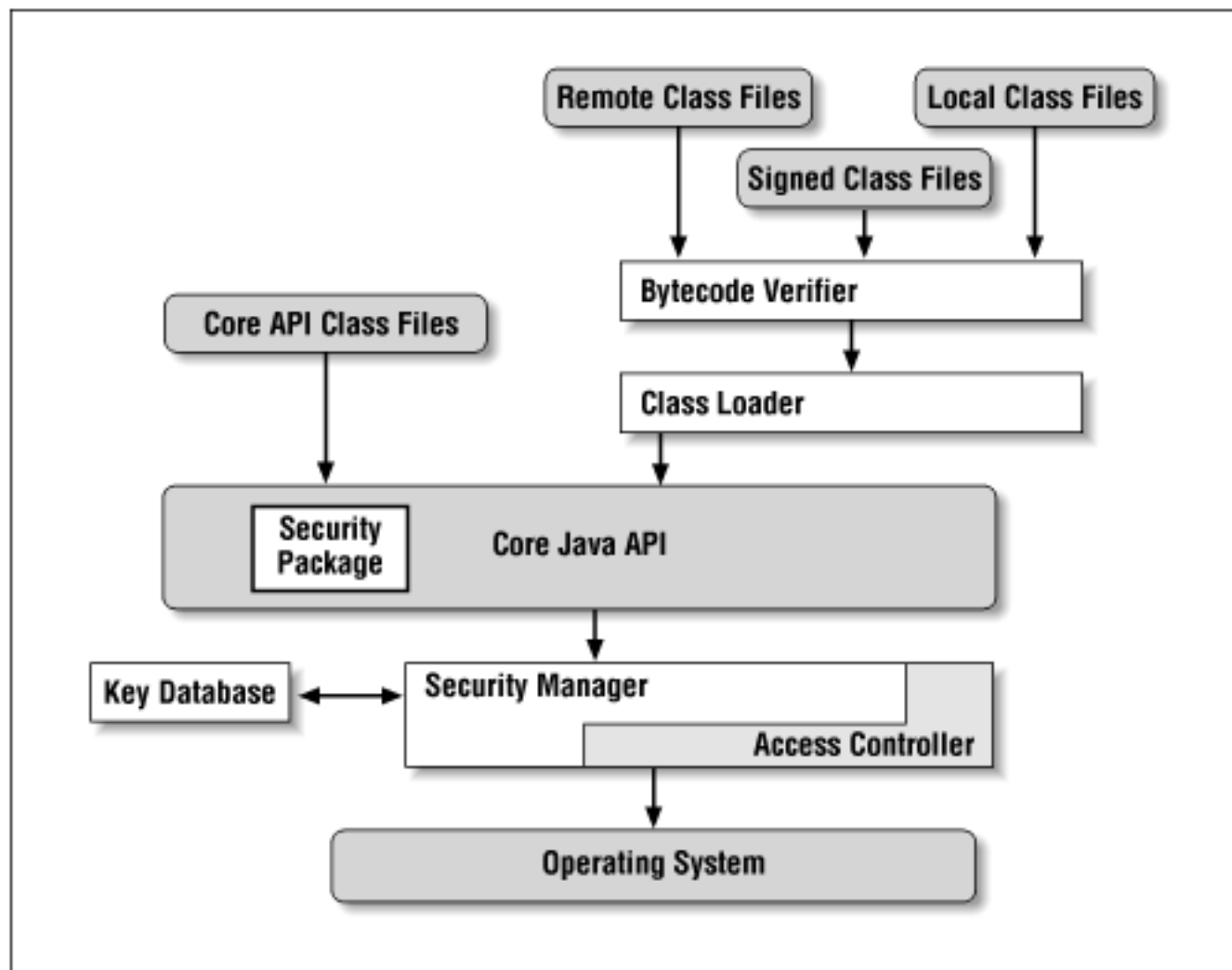
**Secure coding**

En java...

# Java sécurité

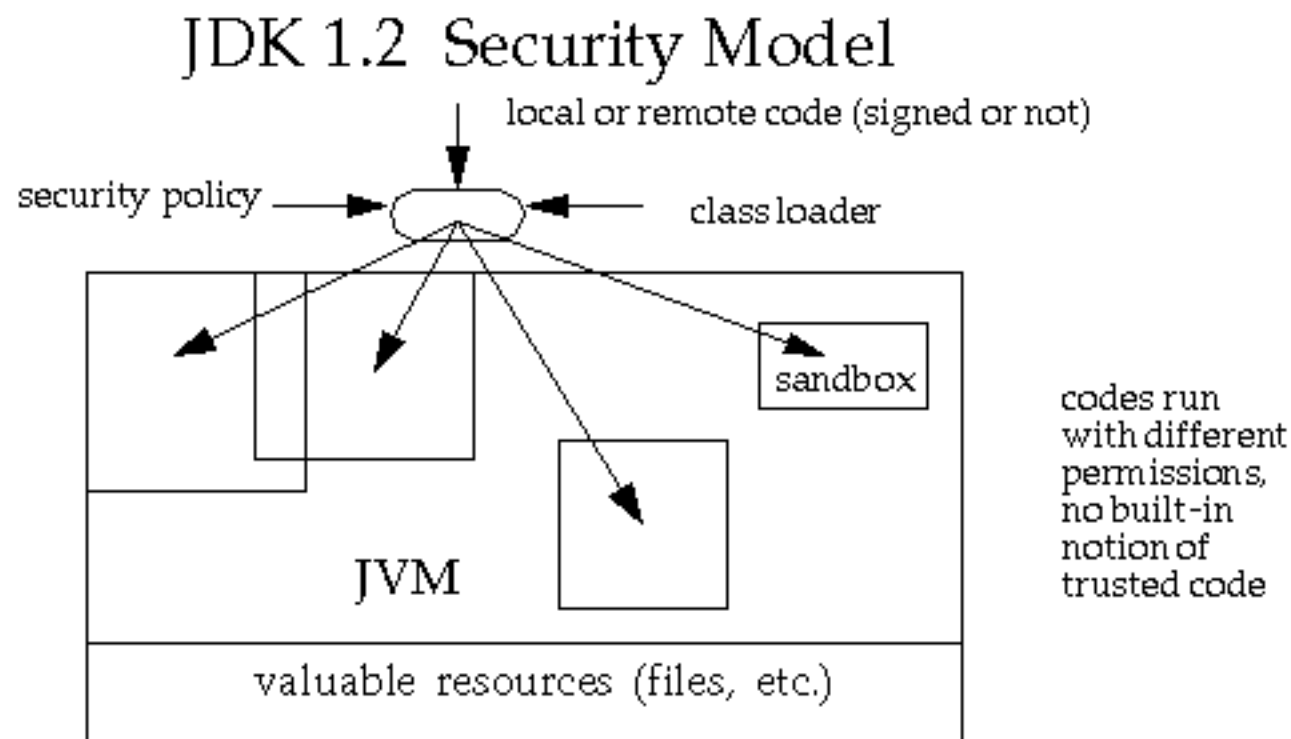
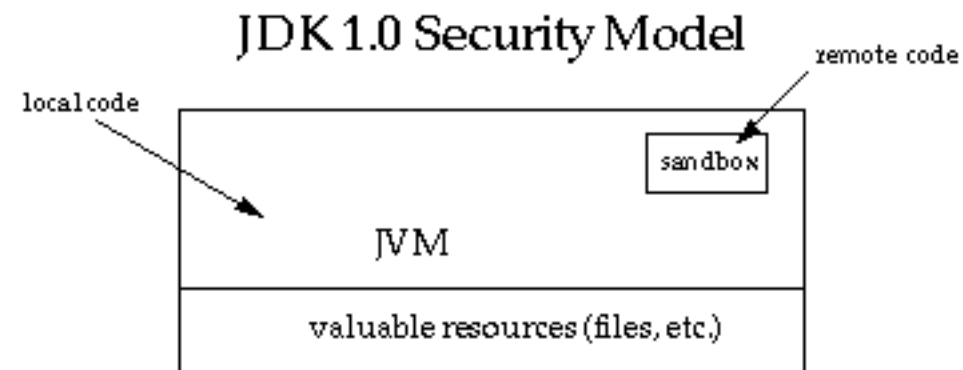
- JVM: vérification du bytecode, vérifications à l'exécution (ex: dépassement des bornes des tableaux), garbage collector, sûreté du typage
- **Sécurité manager: mécanisme de sandbox (bac à sable)** qui isole l'exécution des applications , vérification des signatures du code
- Classes contenant les algorithmes cryptographiques, authentification et protocoles de communication sécurisés.

# Java



# Sécurité: sandbox

modèle initial



modèle jdk1.2

# Exemple Applet

- une applet s'exécute dans une machine virtuelle java
- une applet *non-signée* est limitée:
  - ne peut pas accéder aux ressources du client (ex: fichiers, clipboard, printer ...)
  - ne peut pas se connecter sur des serveurs autres que ceux de son origine
  - ne peut charger des bibliothèques « natives »
  - ne peut pas changer le Security Manager
  - ne peut pas créer de ClassLoader
  - ne peut pas lire certaines propriétés du Système (java.class.path  
java.home user.dir user.home user.name)
  - (JNLP (java Network Launch Protocol) permet d'augmenter ces droits)

# Properties

- Classes Properties
  - extension de Hashtable
  - clé/valeur exemple: `user.home /Users/hf1`
    - `getProperty` `setProperty`
- `System.getProperty(cle)`,  
`System.getProperties()`
  - donne ou modifie la valeur d'une « property »



# Exemple

```
import java.lang.*;
import java.util.Hashtable;
import java.util.Set;
public class GetProps {
    public static void main(String[] args) {
        String s;
        try {
            System.out.println(" os.name property");
            s = System.getProperty("os.name", "");
            System.out.println(" Nom de l'OS : " + s);
            System.out.println("java.version property");
            s = System.getProperty("java.version", "not specified");
            System.out.println(" version de la JVM : " + s);
            System.out.println("user.home property");
            s = System.getProperty("user.home", "not specified");
            System.out.println(" user home directory: " + s);
            System.out.println("user.dir property");
            s = System.getProperty("user.dir", "not specified");
            System.out.println(" user dir directory: " + s);
            System.out.println("java.path property");
            System.out.println(" java path: " + s);
            s = System.getProperty("java.class.path", "not specified");
            System.out.println("java.home property");
            s = System.getProperty("java.home", "not specified");
            System.out.println(" catalogue de la JRE: " + s);
        } catch (Exception e) {
            System.err.println("exception " + e.toString());
            (System.getProperties()).list(System.out);
        }
    }
}
```

# Résultat

os.name property

Nom de l'OS :Mac OS X

java.version property

version de la JVM : 1.8.0

user.home property

user home directory: /Users/hf1

user.dir property

user dir directory: /Users/hf1/Netbeans/java/securite

java.path property

java path: /Users/hf1/Netbeans/java/securite

java.home property

catalogue de la JRE: /Library/Java/JavaVirtualMachines/  
jdk1.8.0.jdk/Contents/Home/jre

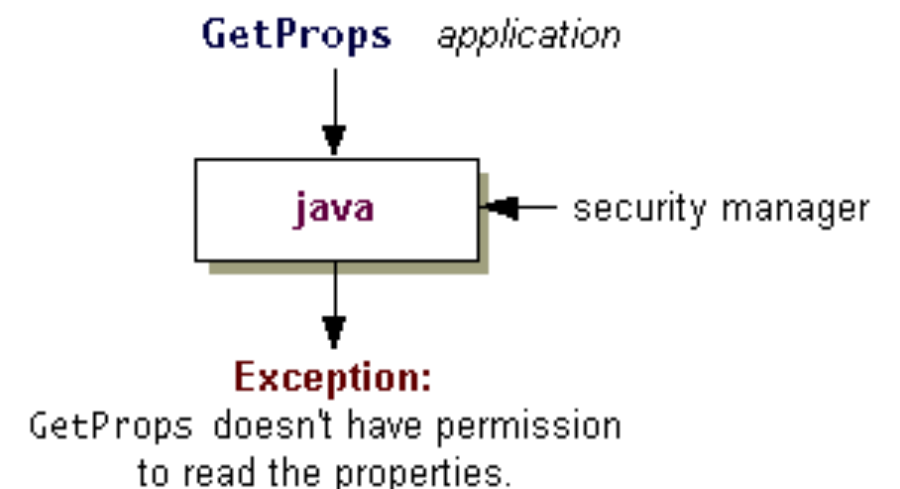
# Sécurité

- sécurité manager (SecurityManager) définit la politique de sécurité pour une application
- par défaut aucun security manager n'est installé: pour l'installer:  
`java -Djava.security.manager ...`

- une action non-autorisée lance l'exception `SecurityException`

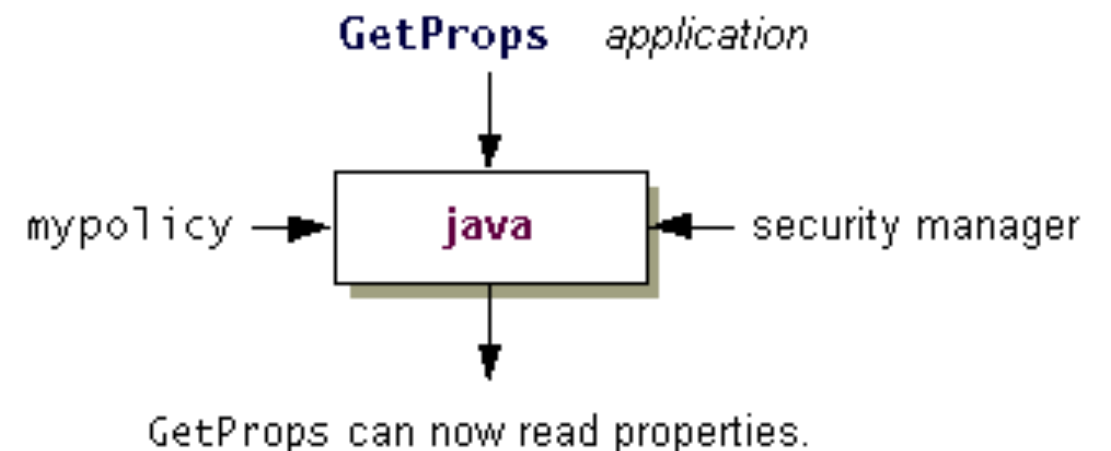
```
SecurityManager sm = System.getSecurityManager();  
    if (sm != null) {  
        sm.checkXXX(argument, . . . )  
    }
```

lance une exception si XXX n'est pas autorisé  
(exemples `checkDelete(fichier)`, `checkExit(status)`)



# Security manager

- par défaut, `user.home` et `java.home` ne sont pas autorisés
- la politique par défaut dans le fichier  
`java.home/lib/security/java.policy`  
`java.home/lib/security/java.security`  
`user.home/.java.policy`
- on peut définir une politique spécifique avec: `policytool`



# défaut: *java.home/lib/security/java.policy*

```
// Standard extensions get all permissions by default

grant codeBase "file:${java.home}/lib/ext/" {
    permission java.security.AllPermission;
};

// default permissions granted to all domains

grant {
    // allows anyone to listen on un-privileged ports
    permission java.net.SocketPermission "localhost:1024-", "listen";

    // "standard" properties that can be read by anyone

    permission java.util.PropertyPermission "java.version", "read";
    permission java.util.PropertyPermission "java.vendor", "read";
    permission java.util.PropertyPermission "java.vendor.url", "read";
    permission java.util.PropertyPermission "java.class.version", "read";
    permission java.util.PropertyPermission "os.name", "read";
    permission java.util.PropertyPermission "os.version", "read";
    permission java.util.PropertyPermission "os.arch", "read";
    permission java.util.PropertyPermission "file.separator", "read";
    permission java.util.PropertyPermission "path.separator", "read";
    permission java.util.PropertyPermission "line.separator", "read";

    permission java.util.PropertyPermission "java.specification.version", "read";
    permission java.util.PropertyPermission "java.specification.vendor", "read";
    permission java.util.PropertyPermission "java.specification.name", "read";

    permission java.util.PropertyPermission "java.vm.specification.version", "read";
    permission java.util.PropertyPermission "java.vm.specification.vendor", "read";
    permission java.util.PropertyPermission "java.vm.specification.name", "read";
    permission java.util.PropertyPermission "java.vm.version", "read";
    permission java.util.PropertyPermission "java.vm.vendor", "read";
    permission java.util.PropertyPermission "java.vm.name", "read";
};
```

```

#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
#
# Default login configuration file
#
#login.config.url.1=file:${user.home}/.java.login.config

# whether or not we allow an extra policy to be passed on the command line
# with -Djava.security.policy=somefile. Comment out this line to disable
# this feature.
policy.allowSystemProperty=true

# whether or not we look into the IdentityScope for trusted Identities
# when encountering a 1.1 signed JAR file. If the identity is found
# and is trusted, we grant it AllPermission.
policy.ignoreIdentityScope=false

#
# Default keystore type.
#
keystore.type=jks
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageAccess unless the
# corresponding RuntimePermission ("accessClassInPackage."+package) has
# been granted.
package.access=sun.,\
               com.sun.xml.internal.,\
               com.sun.imageio.,\
               com.sun.istack.internal.,\...
# List of comma-separated packages that start with or equal this string
# will cause a security exception to be thrown when
# passed to checkPackageDefinition unless the
# corresponding RuntimePermission ("defineClassInPackage."+package) has
# been granted.
#
# by default, none of the class loaders supplied with the JDK call
# checkPackageDefinition.
#
package.definition=sun.,\
                  com.sun.xml.internal.,\
                  com.sun.imageio.,\
                  com.sun.istack.internal.,\...

```

java.security

# résultat de policytool

```
/* AUTOMATICALLY GENERATED ON Wed Feb 18 15:31:26 CET 2015*/  
/* DO NOT EDIT */
```

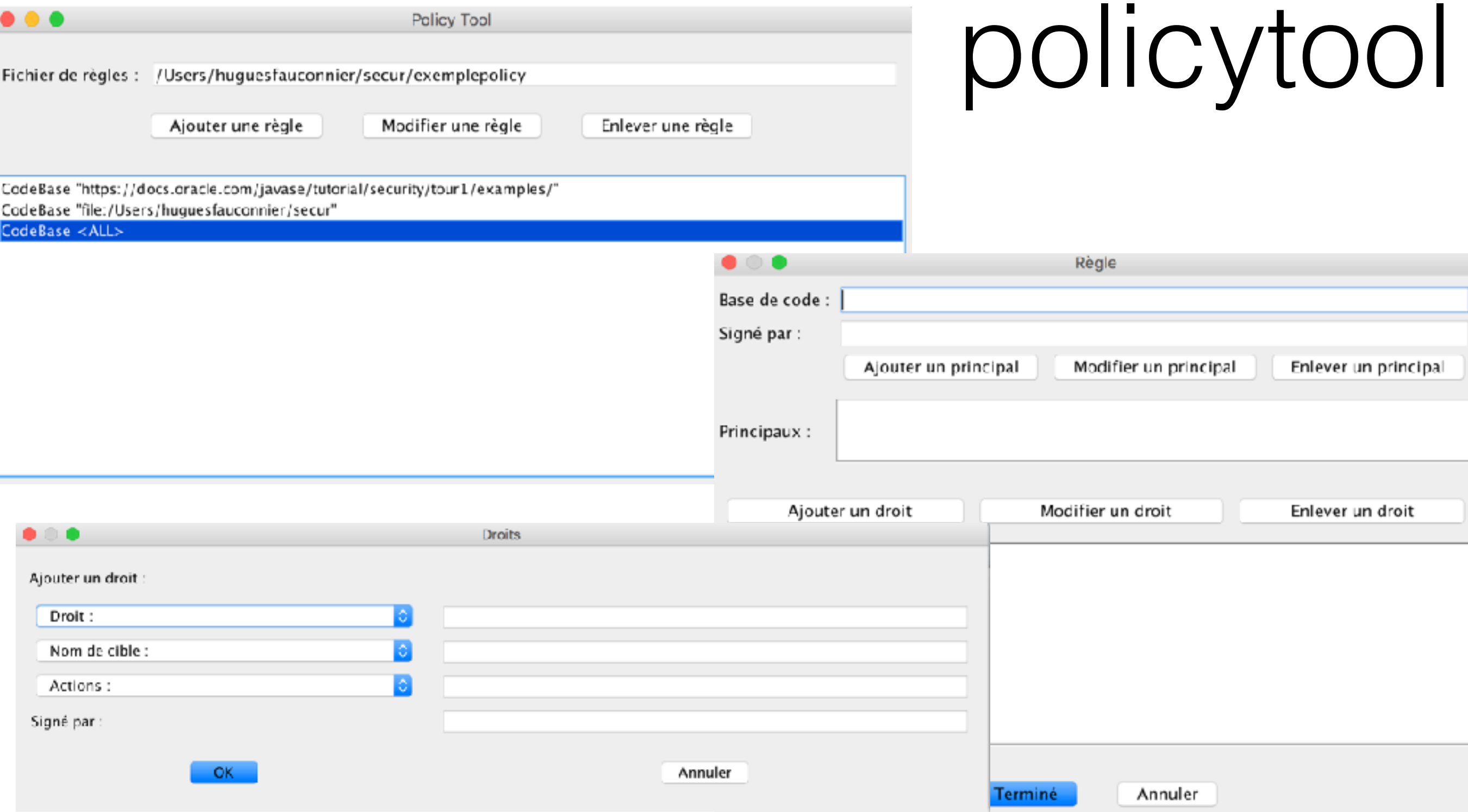
```
grant codeBase "http://docs.oracle.com/javase/tutorial/security/tour1/examples/" {  
};
```

```
grant codeBase "file:/Users/hf1/Netbeans/java/securite/src/securite" {  
    permission java.util.PropertyPermission "user.home", "read";  
    permission java.util.PropertyPermission "java.home", "read";  
};
```

-----

```
java -Djava.security.manager -Djava.security.policy=maPolitique GetProps
```

# policytool



```
java -Djava.security.manager -Djava.security.policy=examplepolicy Appli
```



# politique de sécurité

Contenu:

```
grant codeBase "file:${{java.ext.dirs}}/*" {  
    permission java.security.AllPermission;  
};
```

les fichiers `file:${{java.ext.dirs}}/*` auront la permission `java.security.AllPermission` (toutes les autorisations possibles).

S'il y a un sécurité manager installé, les extensions (jar) dans ces fichiers auront les privilèges nécessaires.

Les codes instances de `PrivilegedAction` en argument de `doPrivileged` aussi.

```
package com.tutorialspoint;

import java.io.FilePermission;
import java.security.AccessControlContext;
import java.security.AccessController;

public class SecurityManagerDemo extends SecurityManager {

    public static void main(String[] args) {
        // le contexte et définir la politique
        AccessControlContext con = AccessController.getContext();
        System.setProperty("java.security.policy", "file:/home/hf/java.policy");
        // créer et utiliser un security manager
        SecurityManagerDemo sm = new SecurityManagerDemo();
        System.setSecurityManager(sm);
        // vérifier l'accès
        sm.checkPermission(new FilePermission("test.txt", "read,write"), con);
        System.out.println("autorisé!");
    }
}
```

```
java.policy:
grant {
    permission java.lang.RuntimePermission "setSecurityManager";
    permission java.lang.RuntimePermission "createSecurityManager";
    permission java.lang.RuntimePermission "usePolicy";
};
```

Exception in thread "main" java.security.AccessControlException: access denied (java.io.FilePermission test.txt read,write)

```

import java.io.*;
class PasswordSecurityManager extends SecurityManager {
    private String password;
    PasswordSecurityManager(String password) {
        super();
        this.password = password;
    }
    private boolean accessOK() {
        int c;
        DataInputStream dis = new DataInputStream(System.in);
        String response;
        System.out.println(" Le password?");
        try {
            response = dis.readLine();
            if (response.equals(password))
                return true;
            else
                return false;
        } catch (IOException e) {
            return false;
        }
    }
    public void checkRead(FileDescriptor filedescriptor) {
        if (!accessOK())
            throw new SecurityException("Raté!");
    }
    public void checkRead(String filename) {
        if (!accessOK())
            throw new SecurityException("No Way!");
    }
    public void checkRead(String filename, Object executionContext) {
        if (!accessOK())
            throw new SecurityException(" C'est foutut!");
    }
    public void checkWrite(FileDescriptor filedescriptor) {
        if (!accessOK())
            throw new SecurityException("Non!");
    }
    public void checkWrite(String filename) {
        if (!accessOK())
            throw new SecurityException("même pas en rêve!");
    }
}

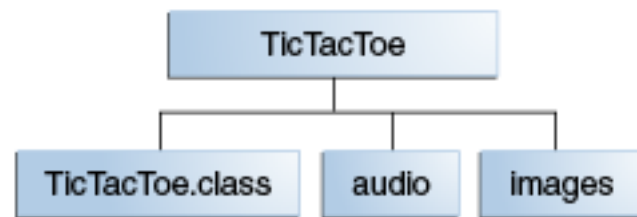
```

Exemple

# fichiers jar

- créer un fichier jar: `jar cf jar-file input-file(s)`
- voir le contenu d'un fichier jar: `jar tf jar-file`
- extraire le contenu: `jar xf jar-file`
- extraire un contenu: `jar xf jar-file archived-file(s)`
- executer: `java -jar app.jar`
- applet:

```
<applet code=AppletClassName.class  
        archive="JarFileName.jar"  
        width=width height=height>  
</applet>
```



# Exemples

- exemples:

```
jar cvf TicTacToe.jar TicTacToe.class \
    audio images
(Créer)
```

```
jar tvf TicTacToe.jar
(contenu)
```

```
jar xf TicTacToe.jar TicTacToe.class \
    images/cross.gif
(eXtraire)
```

```
jar uf TicTacToe.jar images/new.gif
(update)
```

```
(java -jar app.jar exécution)
```

# manifest

- définit les fonctionnalités de l'archive
- création par défaut:  
META-INF/MANIFEST.MF

Manifest-Version: 1.0

Created-By: 1.7.0\_06 (Oracle Corporation)

- `jar cfm jar-file manifest-addition input-file(s)`  
pour changer le contenu du manifest

# MANIFEST

## Exemple:

Manifest-Version: 1.0

Ant-Version: Apache Ant 1.9.4

Created-By: 1.8.0-b132 (Oracle Corporation)

Class-Path:

X-COMMENT: Main-Class will be added automatically by build

Main-Class: securite.GetProps

Main-Class est le point d'entrée pour exécuter l'application

On peut aussi changer le point d'entrée.

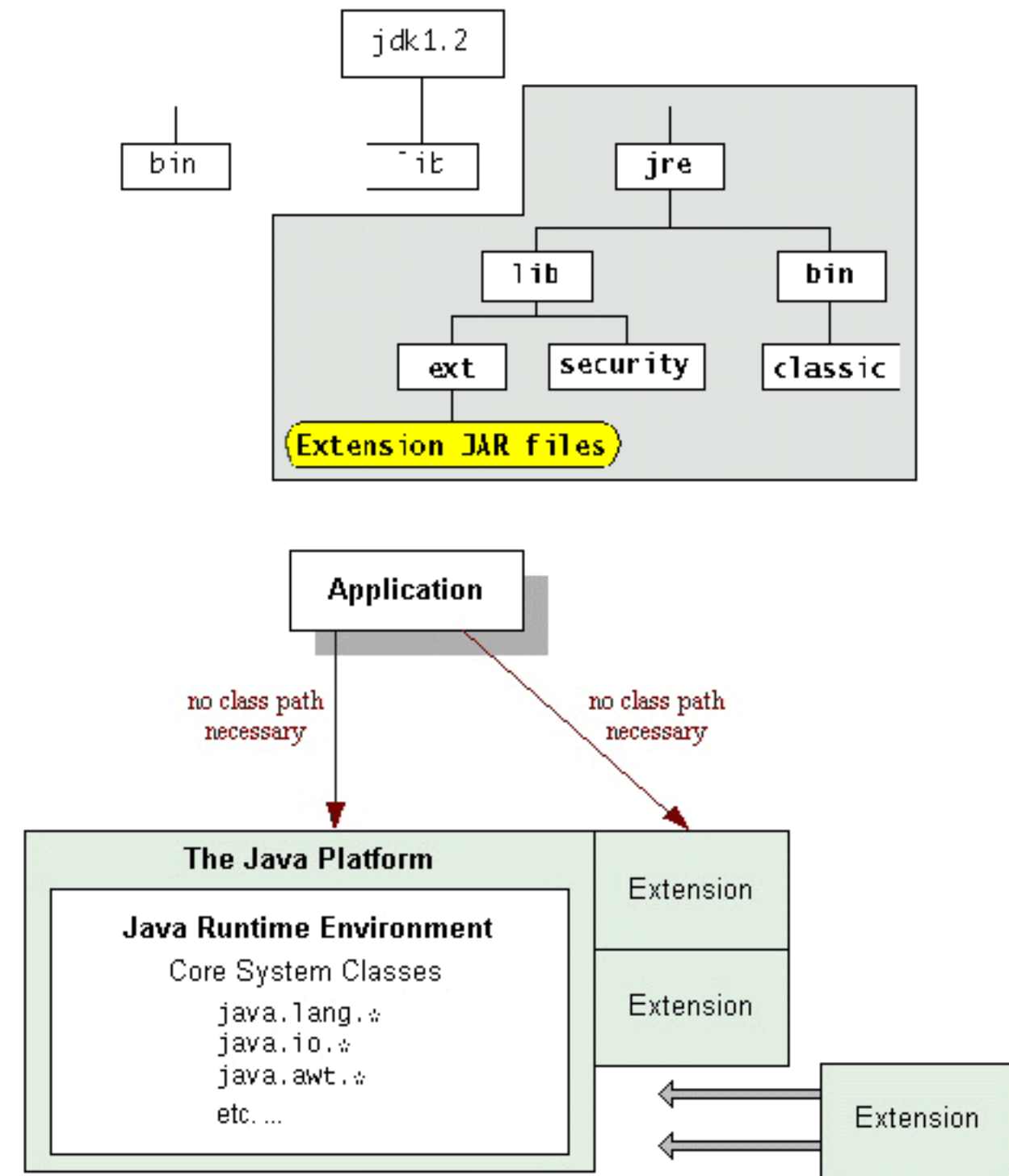
Class-Path: pour contenir des extensions .jar

```
jar cfe app.jar MyApp MyApp.class
```

```
jar cfe Main.jar foo.Main foo/Main.class
```

# Extensions

- structure du jdk
- ext contient des (.jar) extensions que l'on peut ajouter à l'environnement de la JVM
- (alternative:  
`java -classpath ext.jar appli`  
)





# Extensions et sécurité

```
import java.io.*;
import java.security.*;

public final class MaClass {
    public static void
        maFonc(final Param r) {
        AccessController.
            doPrivileged(new PrivilegedAction() {
                public Object run() {...
                    // nécessite des privilèges
                }
            });
    }
}
```

code dans la méthode run d'un objet

java.security.PrivilegedAction

méthode doPrivileged appliquée à la PrivilegedAction rend se code privilégié

(sinon les permissions sont les permissions minimales de la chaîne des appels)

# AccessController

- AccessController
- méthodes:
  - checkPermission

```
FilePermission perm = new FilePermission("/temp/testFile", "read");  
AccessController.checkPermission(perm);
```

- doPrivileged

```
somemethod() {  
    ...normal code here...  
    AccessController.doPrivileged(new PrivilegedAction<Void>() {  
        public Void run() {  
            // privileged code goes here, for example:  
            System.loadLibrary("awt");  
            return null; // nothing to return  
        }  
    });  
    ...normal code here...  
}
```

```

public void doStuff() {

    try {
        /* exception si pas de permission pour l'appelant */
        System.out.println(System.getProperty("java.home"));
    } catch (Exception e1) {
        System.out.println(e1.getMessage());
    }
    AccessController.doPrivileged(new PrivilegedAction<Boolean>() {
        public Boolean run() {
            try {
                /*
                 * ok si la classe a la permission même
                 * si l'appelant ne l'a pas
                 */
                System.out.println(System.getProperty("java.home"));
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }

            return Boolean.TRUE;
        }
    })
}

```

doPrivileged

```

grant codeBase "file:/home/somebody/classb.jar" {
    permission java.util.PropertyPermission "java.home", "read";
};

```

doStuff() est défini dans classb et appelé par classa