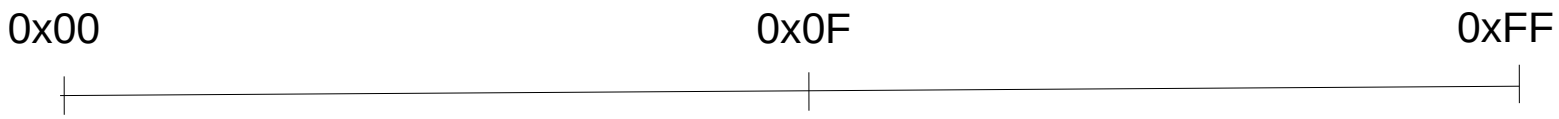
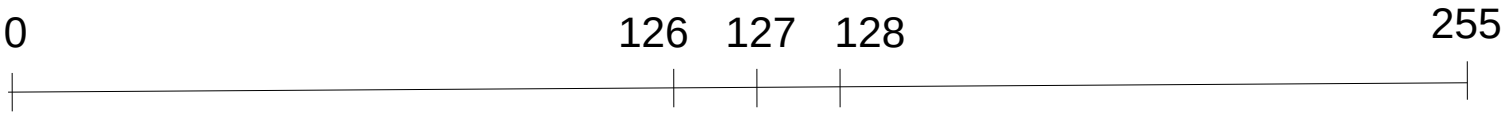
Two’s complement representation

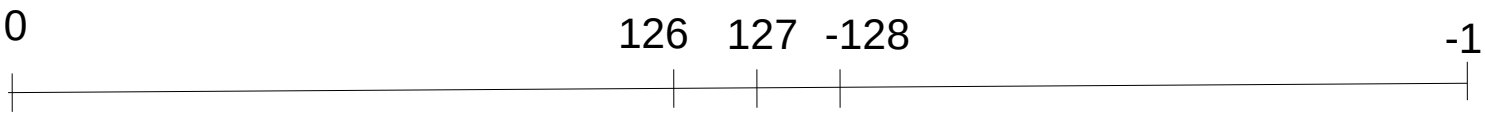
hexadecimal



unsigned



signed



In case of wrap-around :

- signed : 255+1=0
- unsigned : 127+1=-128 and -128+1 = 127

Exercise 1- explanation

```
int offset, len ;  // signed integers
...
/* first check that both offset and len are positives */
if (offset < 0 || len <= 0)
    return -EINVAL;
/* if offset + len exceeds the MAXSIZE threshold, or in case of overflow,
    return an error code */
```

offset and len are both signed positive values and signed integer overflow is an undefined behavior

=> offset + len <0 is not expected to happen, this check is removed by the optimizer !

```
if ((offset + len > MAXSIZE) || (offset + len < 0)
    return -EFBIG // offset + len does overflow
/* assume from now on that len + offset did not overflow ... */
```

offset + len can be a negative value ...

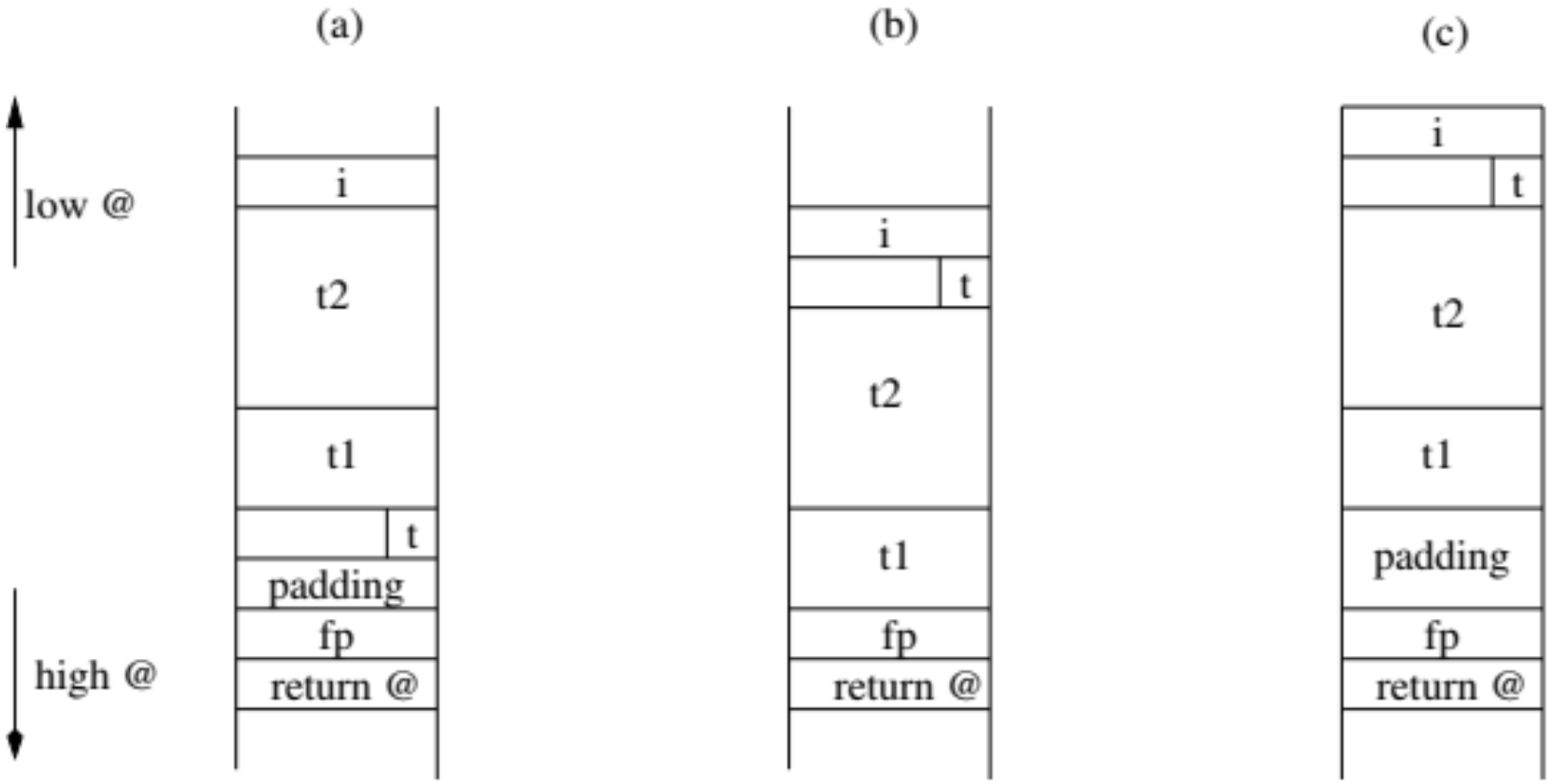
Exercise 1- possible corrections

1) See the corresponding CERT secure coding pattern

2) used unsigned integers
(no undefined behaviors, always wrap-arround)

3) use compiler options to enfore wrap-aroud :
-fno-strict-overflow and *-f-wrapv*

Exercise 2


(a)
(b)
(c)
low @
high @
i
t2
t1
t
padding
fp
return @
i
t
t2
t1
fp
return @
i
t
t2
t1
padding
fp
return @

Exercise 3

```
typedef struct {void (*f)(void);} st;
void nothing (){ printf("Nothing\n"); }

int main(int argc , char * argv [])
{ st *p1;
 char *p2;
 p1=(st*) malloc(sizeof(st));
 p1 ->f=& nothing;
 free(p1);
 p2=malloc(strlen(argv [1]));     – p2 may point to the memory cell just freed
 strcpy(p2 ,argv [1]);            -- this cell is initialized with a user input
 p1 ->f();
return 0;}
```

=> arbitrary code execution ! (see next Lab)

Assigning pointers to NULL when they are freed ?

- Apply this solution to the previous example
- Explain why this solution may not work in case of compiler optimization
  - Assignment p1=null may be suppressed if p1 is not re-used (but one of its aliases is)
- Explain why this solution is not complete
  - Pointer aliases are not set to NULL
- Propose a more complete solution
  - Use a static alias detection analysis (pb : undecidable, over-approximation)
  - Runtime garbage collection (pb : efficiency issues?)

Exercise 4

```
$userName = $_POST["user"];
$command = 'ls -l /home/' . $userName;
system($command);
```

Explain and correct the security weaknesses of this code ?
Possible input: ;rm -rf /

=> assigns to $command:

ls -l /home/;rm -rf /

Exercise 4 - correction

- **Black listing** : check for occurrences of dangerous characters (e.g, ; | &)

- White listing with a regular expression
- ^[a-z0-9_-]{3,15}$

- https://unix.stackexchange.com/questions/157426/what-is-the-regex-to-validate-linux-users

Exercise 4 - Example

```
static bool is_valid_name (const char *name)
{ /* * User/group names must match [a-z_][a-z0-9_-]*[$] */
  if (('\0' == *name) ||
  !((('a' <= *name) && ('z' >= *name)) || ('_' == *name)))  {
  return false;
}
while ('\0' != *++name) {
if (!(( ('a' <= *name) && ('z' >= *name) ) ||
    ( ('0' <= *name) && ('9' >= *name) ) || ('_' == *name) ||
    ('-' == *name) || ( ('$' == *name) && ('\0' == *(name + 1)) ) )) {
    return false; } }
return true; }
```

(assuming the input size has been checked beforehand)

Exercise 5 question 1

- os.mkdir(*path*[, *mode*]) : Create a directory named *path* with numeric mode *mode*. The default *mode* is 0777 (octal). If the directory already exists, OSError is raised.

- Possible caveats:
  - mkdir is a potentialy dangerous operation
  - Defaults permissions are not necessarily known
  - 0777 gives very liberal permissions (**drwxrwxrwx**)

Exercise 5 question 2

What is the security issue in this code ?

```
def makeNewUserDir(username):
  if invalidUsername(username):
  #avoid CWE-22 and CWE-78
  print('Usernames cannot contain invalid characters')
  return False
try:
  raisePrivileges()
  os.mkdir('/home/' + username)
  lowerPrivileges()
except OSError:
  print('Unable to create new user directory for user:' + username)
  return False
return True
```

- May end with high priviledges set
- Beware of the permissions

Exercise 5 question 3

https://cwe.mitre.org/data/definitions/732.html

```
function createUserDir($username){
$path = '/home/'.$username;
if(!mkdir($path)){ return false;}
if(!chown($path,$username)){rmdir($path); return false;}
return true;}
```

- the directory is created with permissions 0777
- changing the owner does not change the rights (any user may still read/write/execute in the directory)
- no verification of the username

chown()

http://pubs.opengroup.org/onlinepubs/7908799/xsh/chown.html

NAME

chown - change owner and group of a file

SYNOPSIS

#include <sys/types.h>
#include <unistd.h>
int chown(const char *_path_, uid_t _owner_, gid_t _group_);

RETURN VALUE

Upon successful completion, 0 is returned. Otherwise, -1 is returned and *errno* is set to indicate the error. If -1 is returned, no changes are made in the user ID and group ID of the file.

ERRORS

The *chown()* function will fail if:
[EACCES] Search permission is denied on a component of the path prefix.
[ELOOP] Too many symbolic links were encountered in resolving *path*.
[ENAMETOOLONG] The length of the *path* argument exceeds {PATH_MAX} or a pathname component is longer than {NAME_MAX}.
[ENOTDIR] A component of the path prefix is not a directory.
[ENOENT] A component of *path* does not name an existing file or *path* is an empty string.
[EPERM] The effective user ID does not match the owner of the file, or the calling process does not have appropriate privileges.
[EROFS] The named file resides on a read-only file system.

Exercise 6

```
int get_and_verify_password(char *real_password) {
   int  result;
   char *user_password[64];
  get_password_from_user_somehow(user_password, sizeof(user_password));
  result = !strcmp(user_password, real_password);
  memset(user_password, 0, strlen(user_password));
  return result;
}
```

https://www.safaribooksonline.com/library/view/secure-programming-cookbook/0596003943/ch13s02.html

Declare password as *volatile*, encode memset operation by hand, use special functions like SecureZeroMemory() (on Windows)