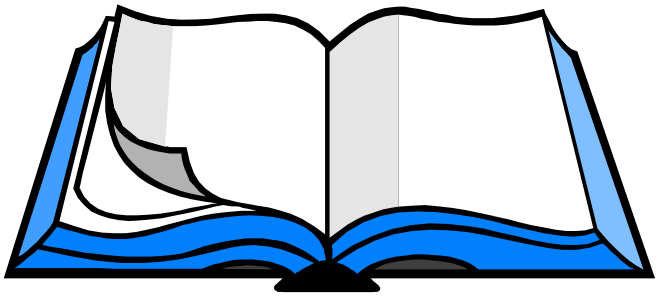
IUT de Lannion **Ludovic Liétard**
Département Informatique
Semestre 3

R3.02 Développement efficace

Support de cours
(La file dynamique)



1. La structure de file

1.1 Définition

1.2 Représentation dynamique d'une file

1.1 Définition

Une file se caractérise par la discipline :

« premier entré, premier sorti »
(First In, First Out : FIFO)

L'ajout et la suppression se font aux extrémités opposées de la file.

On parlera de queue de file (le dernier élément entré) et la tête de file (le premier élément entré = le prochain à sortir).

Basiquement, on ne dispose que de cinq opérations :

Est_Vide :
qui délivre vrai si la file est vide.

Initialisation :
Pour créer une file vide.

Ajout :
qui ajoute un élément (en queue).

Retrait :
qui retire un élément (celui en tête).

ObtenirTête :
qui délivre l'élément en tête de file.

On les implémente par des primitives.
.

1.2 Représentation dynamique d'une file

Les éléments successifs sont chainés entre eux (i.e. sont reliés par des pointeurs). Une file d'entiers est définie par :

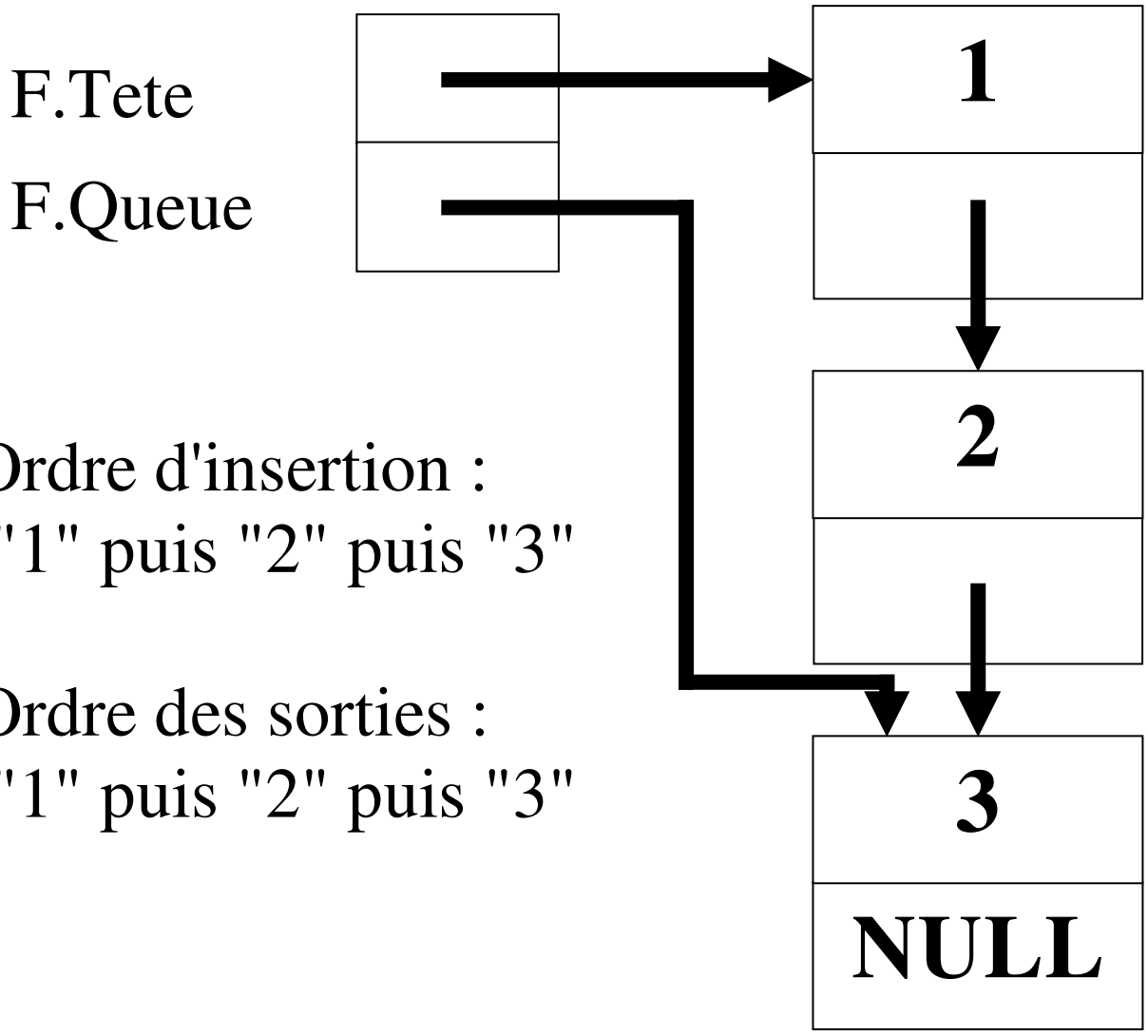
```
typedef struct Elem{
        int val ;
        struct Elem * svt;
} element;

typedef struct{
        element * Queue;
        element * Tete;
} fileD;
```

File vide :

F.Tete **NULL**

F.Queue **NULL**

File non vide (avec 3 éléments):



Ordre d'insertion :
"1" puis "2" puis "3"

Ordre des sorties :
"1" puis "2" puis "3"

```
void Initialisation(fileD * F){
  F->Queue=NULL;
  F->Tete=NULL;
}


int Est_Vide(fileD F){
  if ((F.Queue==NULL) &&
                          (F.Tete==NULL)){
    return 1;
  }else{
    return 0;
  }
}
```

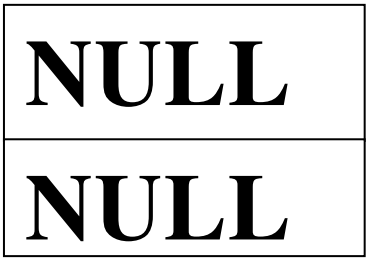
Pour l'ajout, deux cas sont à distinguer :

Cas de la file vide :

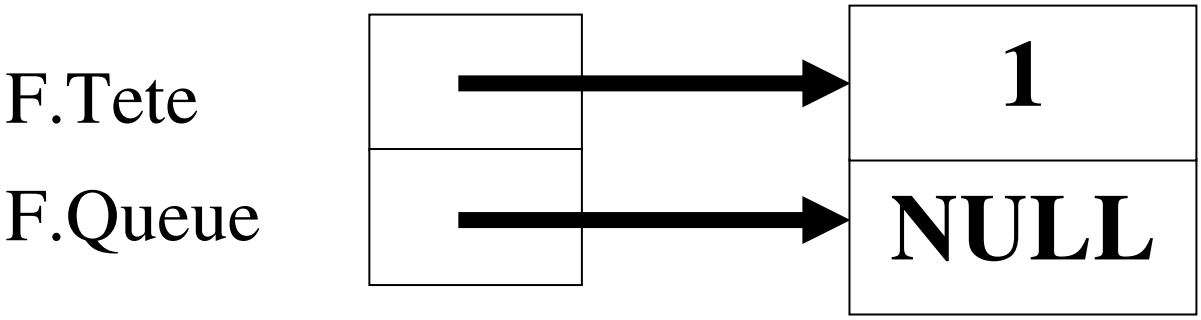
Avant ajout :

F.Tete

F.Queue



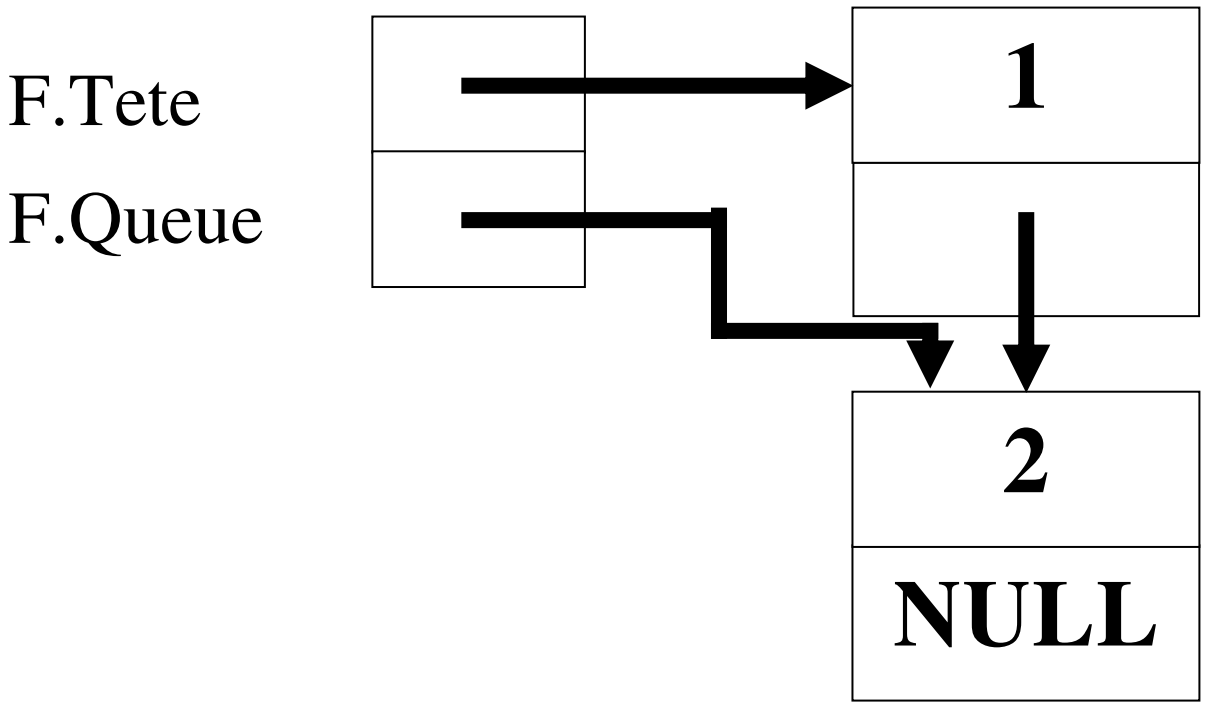
Après ajout :



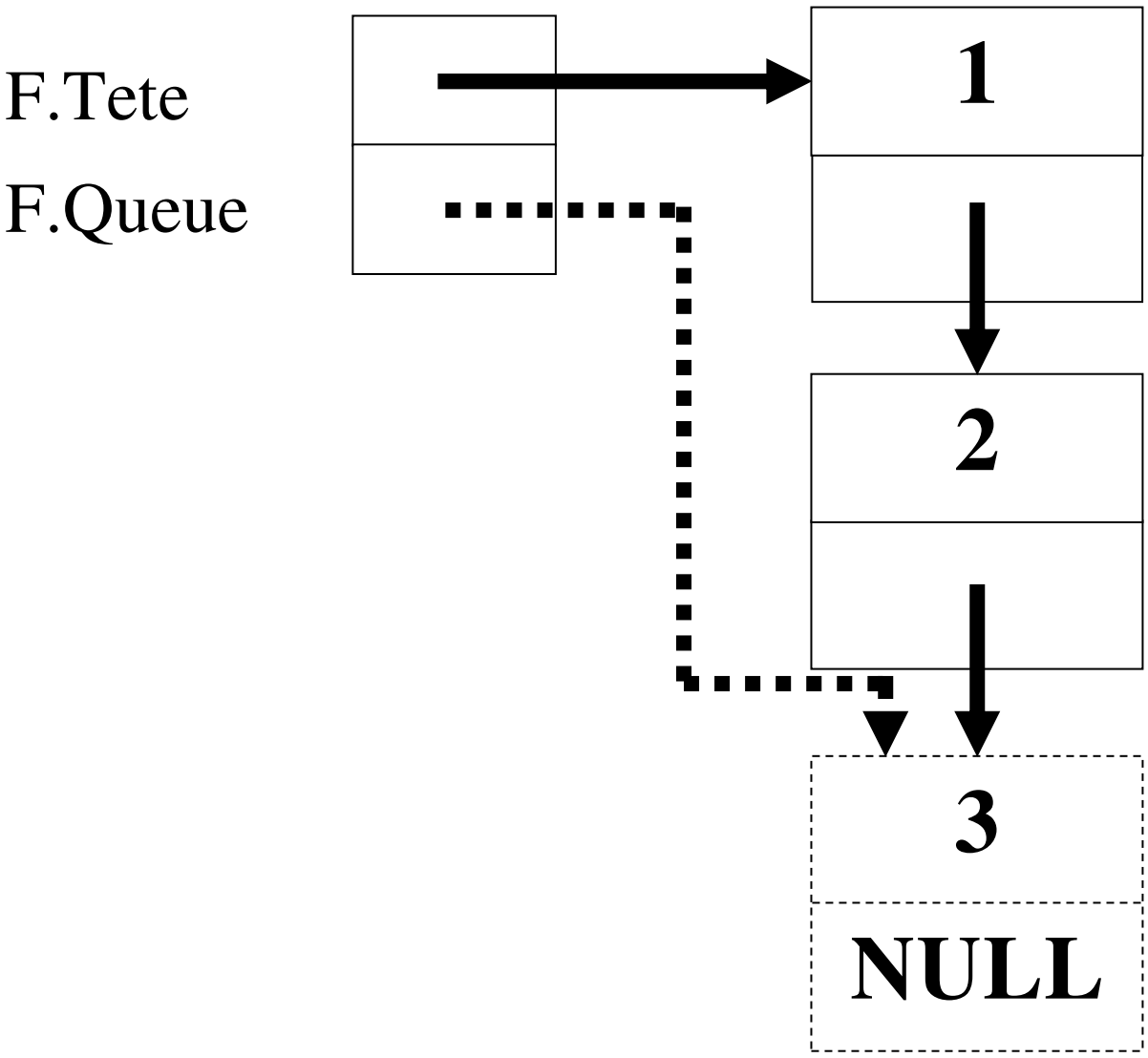
LES DEUX POINTEURS SONT AFFECTES.

Cas de la file non vide :

Avant ajout :



Après ajout :



L'AJOUT NE MODIFIE QUE LE POINTEUR DE QUEUE

```
void Ajout(fileD *F,int valeur){
 element *p;

 p=(element *)
            malloc(sizeof(element));
 p->val=valeur;
 p->svt=NULL;

 if ((F->Queue==NULL) &&
                    (F->Tete==NULL)){
   F->Queue=p;
   F->Tete=p;
 }else{
   F->Queue->svt=p;
   F->Queue=p;
 }
}
```

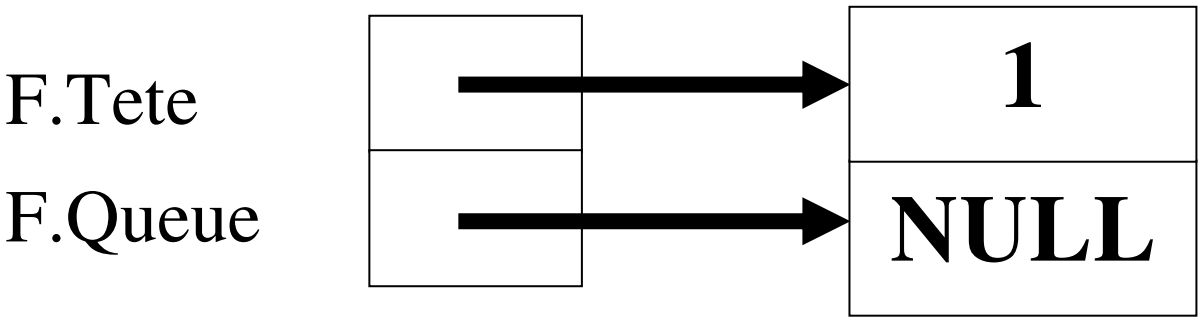
Pour le retrait, deux cas sont à considérer :

1) File ne contenant qu'un seul élément

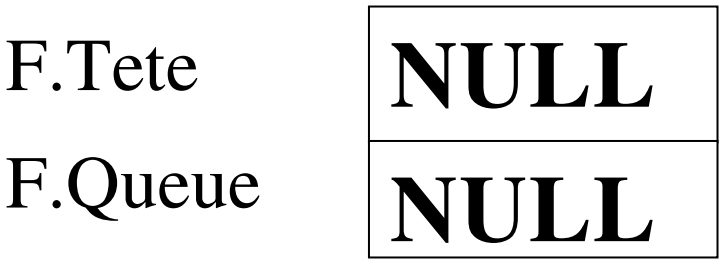
2) File à plusieurs éléments

Cas de la file avec un seul élément :

Avant suppression :



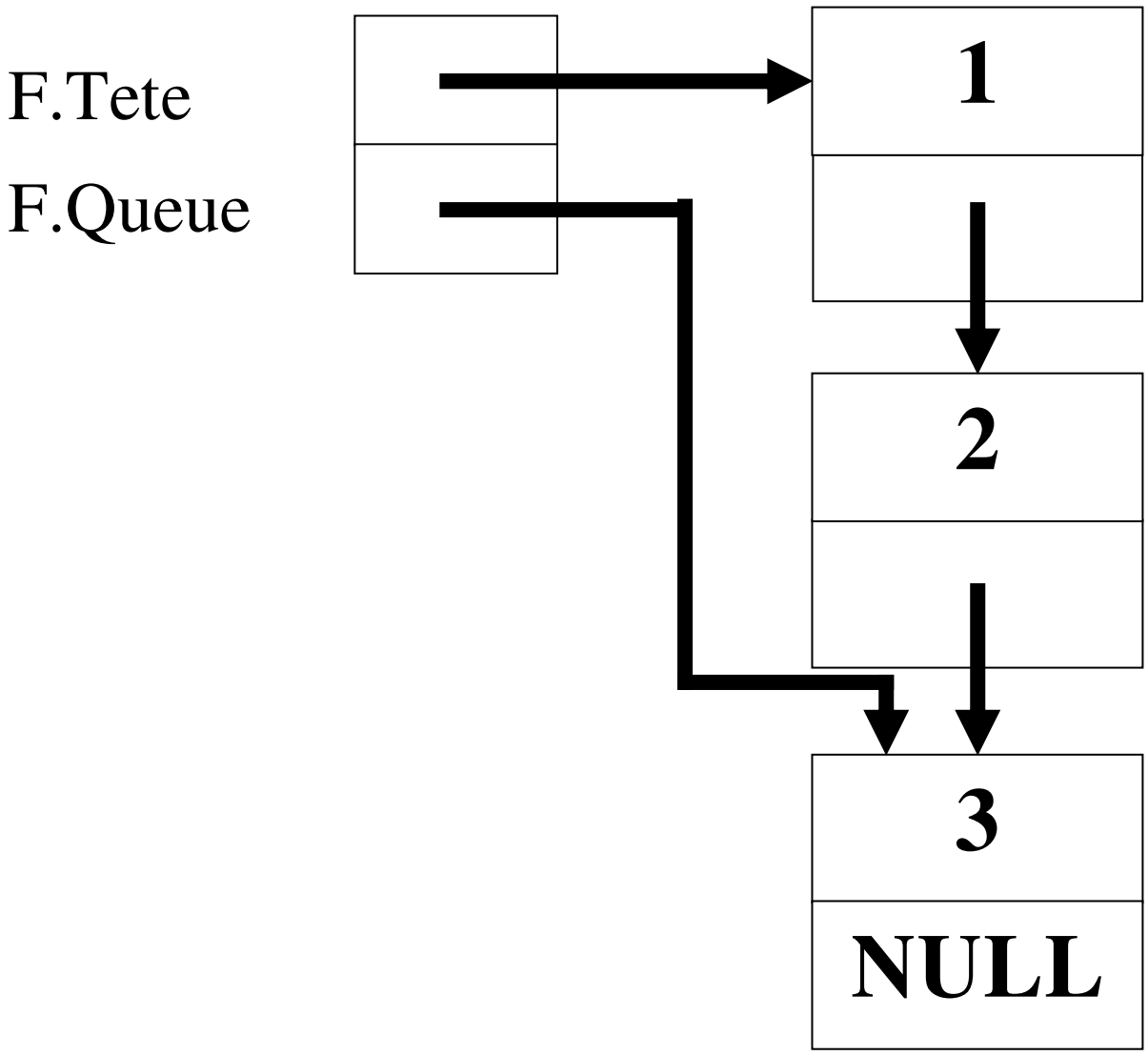
Après suppression :



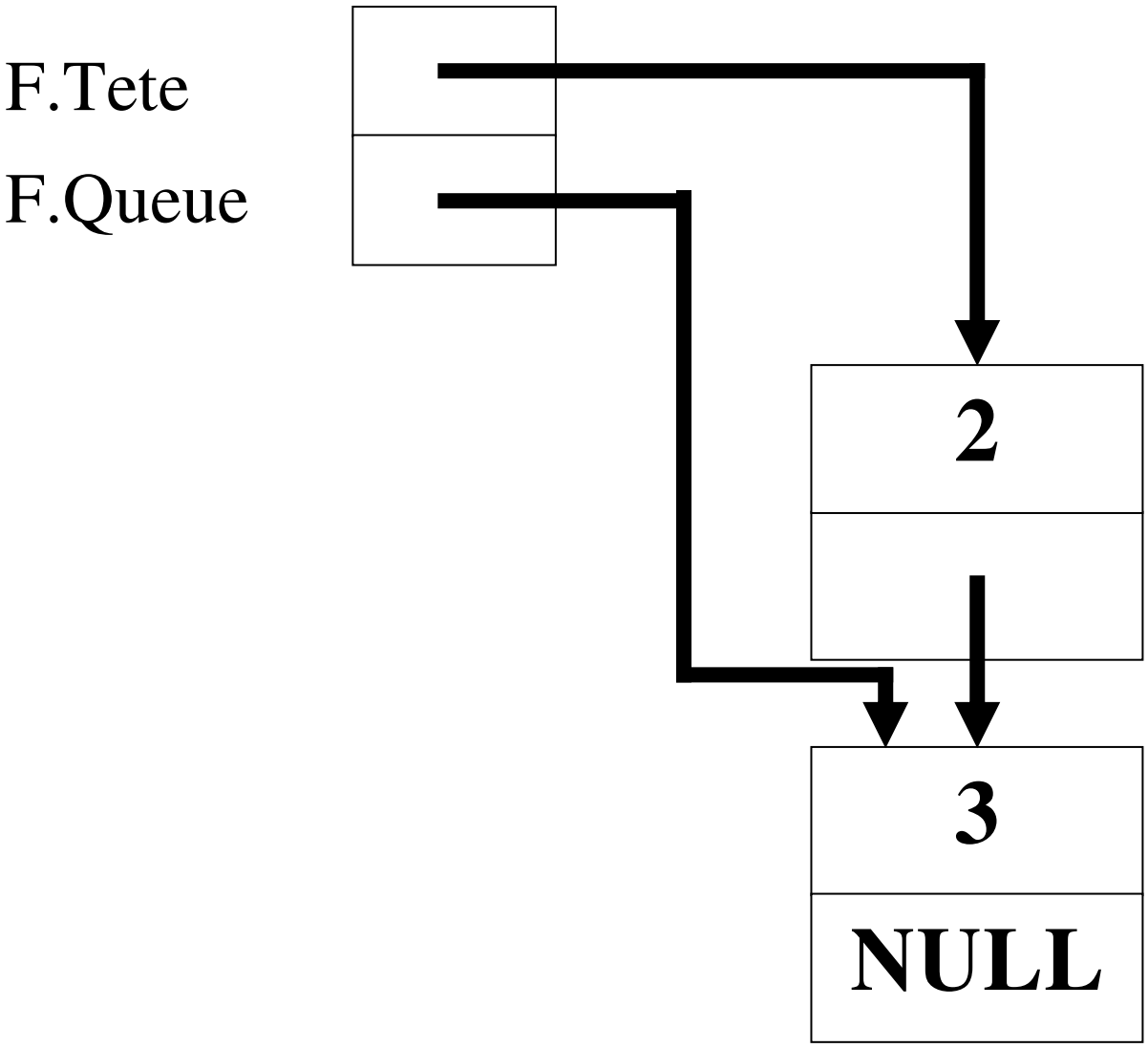
LES DEUX POINTEURS SONT AFFECTES.

Cas de la file avec plusieurs éléments :

Avant suppression :



Après suppression :



LA SUPPRESSION NE MODIFIE QUE LE POINTEUR DE TETE

```
void Retrait(fileD *F){
  element *p;
  if (F->Tete==F->Queue){
    p=F->Tete;
    F->Tete=NULL;
    F->Queue=NULL;
    free(p);
  }else{
    p=F->Tete;
    F->Tete=F->Tete->svt;
    free(p);
  }
}
```

```
int ObtenirTete(fileD F){
  return F.Tete->val;
}
```