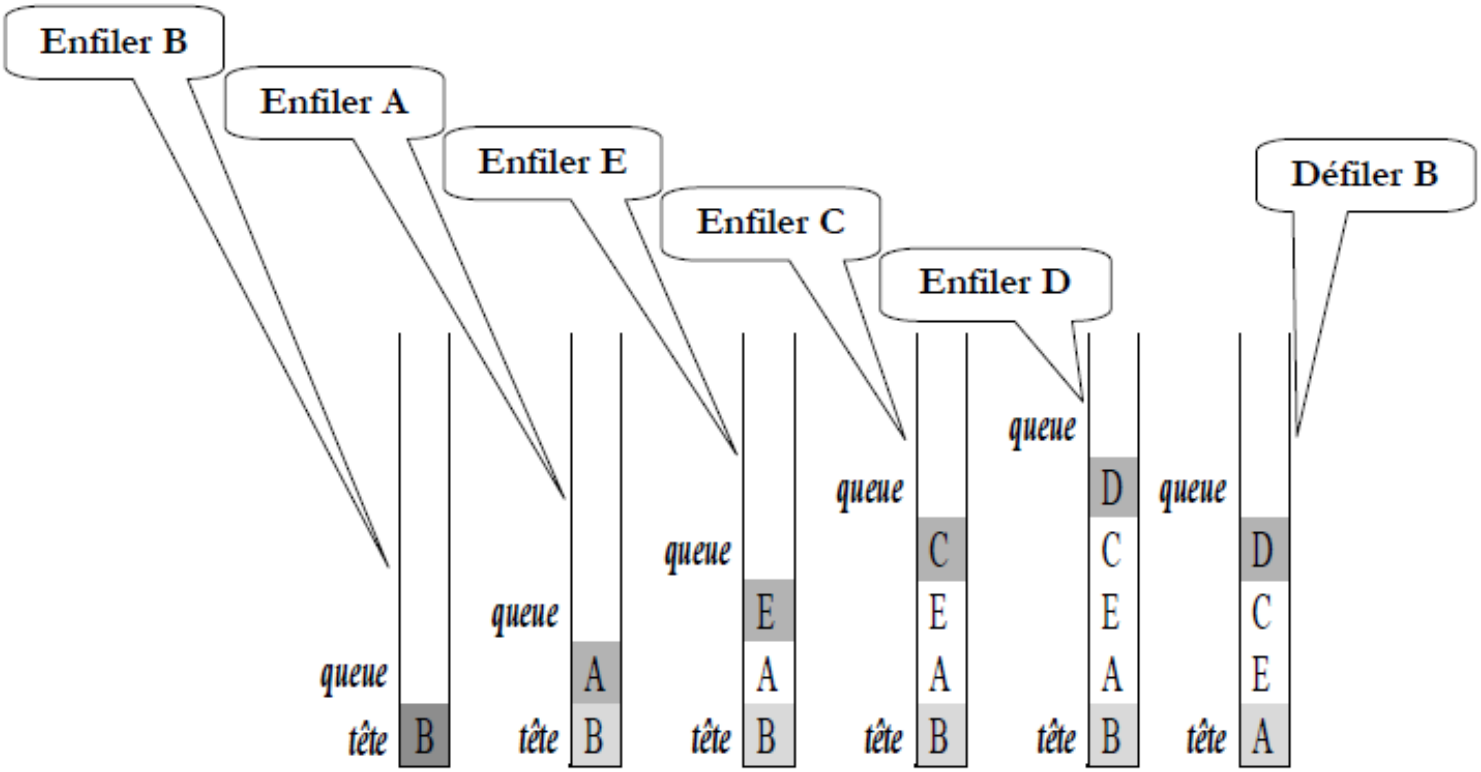
Structures de données linéaires

Files

Notion de File (Queue)

- Les files sont très utilisées en informatique
- Notion intuitive :
  - File d'attente à un guichet, file de documents à imprimer, ...
- Une file est une structure linéaire permettant de stocker et de restaurer des données selon un ordre FIFO (First In, First Out ou « premier entré, premier sorti »)
- Dans une file :
  - Les insertions (enfilements) se font à une extrémité appelée queue de la file et les suppressions (défilements) se font à l'autre extrémité appelée tête de la file.

Exemple de file



Type Abstrait file

Type file
Utilise Élément, Booléen
Opérations
file_vide : $\rightarrow$ file
est_vide : file $\rightarrow$ Booléen
enfiler : file $\times$ Élément $\rightarrow$ file
défiler : file $\rightarrow$ file
tête : file $\rightarrow$ Élément
Préconditions
défiler(f) est-défini-ssi est_vide(f) = faux
tête(f) est-défini-ssi est_vide(f) = faux
Axiomes
Soit, e : Element, f : file
est_vide(file_vide) = vrai
est_vide(enfiler(f,e)) = faux
si est_vide(f) = vrai alors tête(enfiler(f,e)) = e
si est_vide(f) = faux alors tête(enfiler(f,e)) = tête(f)
si est_vide(f) = vrai alors défiler(enfiler(f,e)) = file_vide
si est_vide(f) = faux
alors défiler(enfiler(f,e)) = enfiler(défiler(f),e)

Opérations sur une file

- file_vide : $\rightarrow$ file
  - opération d'initialisation ; la file créée est vide
- est_vide : file $\rightarrow$ Booléen
  - teste si file vide ou non
- tête : file $\rightarrow$ Élément
  - permet de consulter l'élément situé en tête de file ; n'a pas de sens si file vide
- enfiler : file $\times$ Élément $\rightarrow$ file
  - ajoute un élément dans la file
- défiler : file $\rightarrow$ file
  - enlève l'élément situé en tête de la file ; n'a pas de sens si file vide

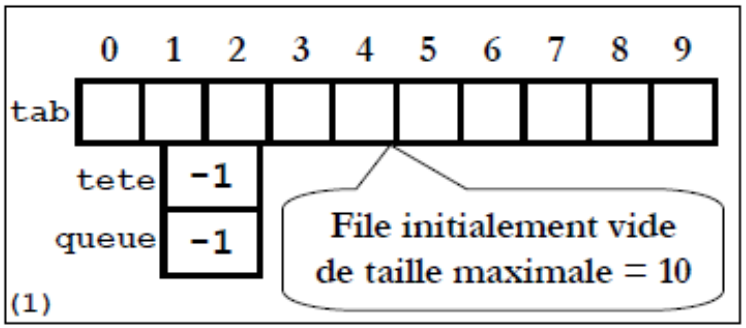
Représentation d'une file

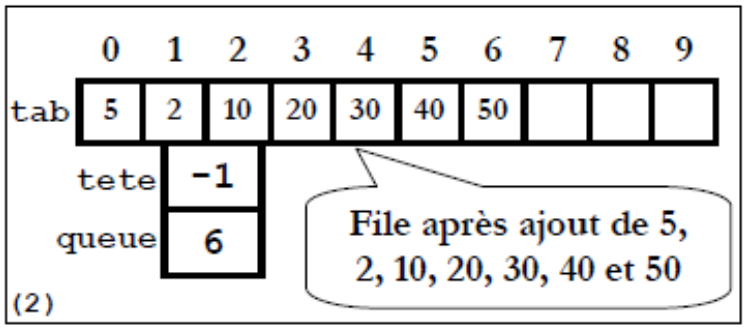
- Représentation contiguë (par tableau) :
  - Les éléments de la file sont rangés dans un tableau
  - Deux entiers représentent respectivement les positions de la tête et de la queue de la file
- Représentation chaînée (par pointeurs) :
  - Les éléments de la file sont chaînés entre eux
  - Un pointeur sur le premier élément désigne la file et représente la tête de cette file
  - Un pointeur sur le dernier élément représente la queue de file
  - Une file vide est représentée par le pointeur NULL

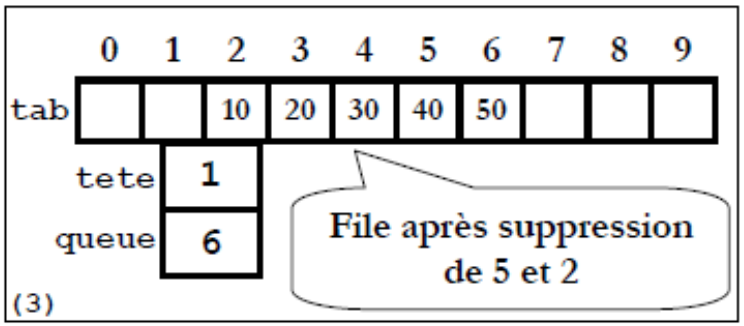
Représentation Contiguë d'une File (par tableau simple)

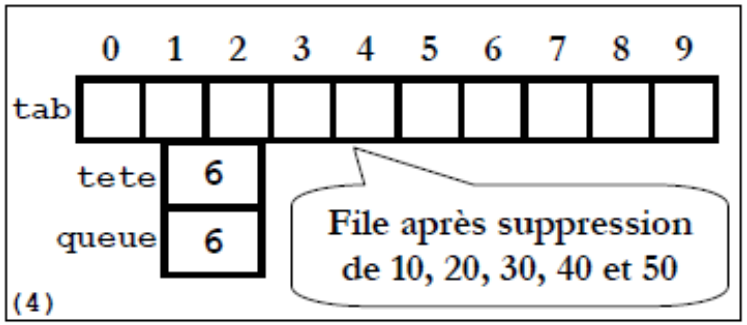
- tête de file : position précédant premier élément
- queue de file : position du dernier élément
- Initialisation : tête $\leftarrow$ queue $\leftarrow$ -1
- Inconvénient : on ne peut plus ajouter des éléments dans la file, alors qu'elle n'est pas pleine !

Représentation Contiguë d'une File (par tableau simple)

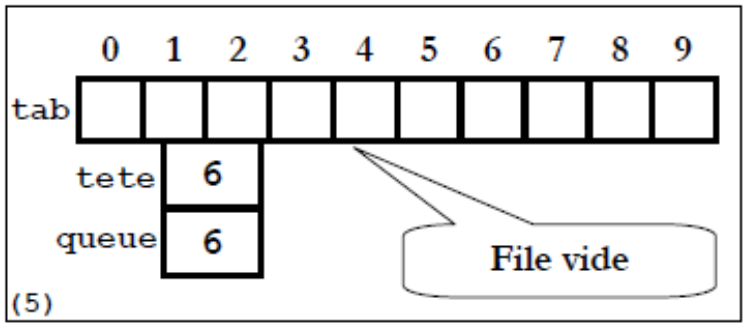


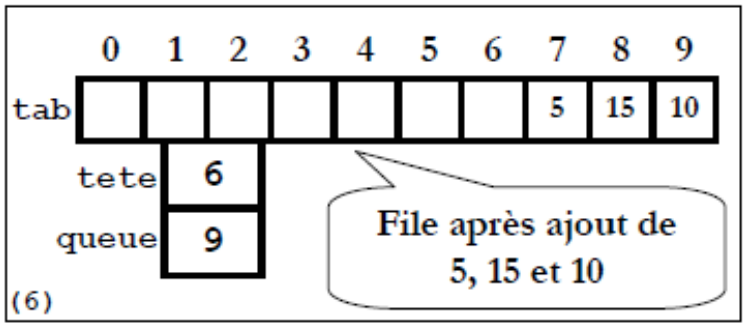


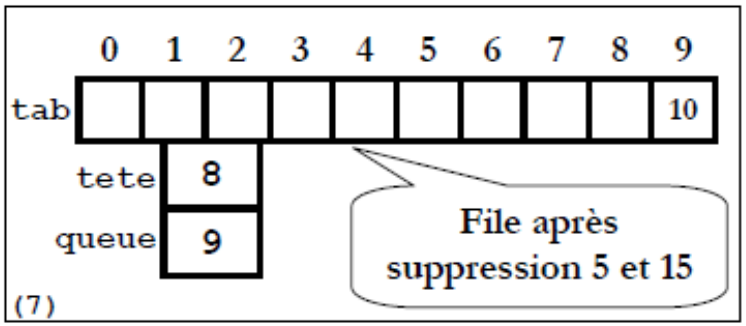


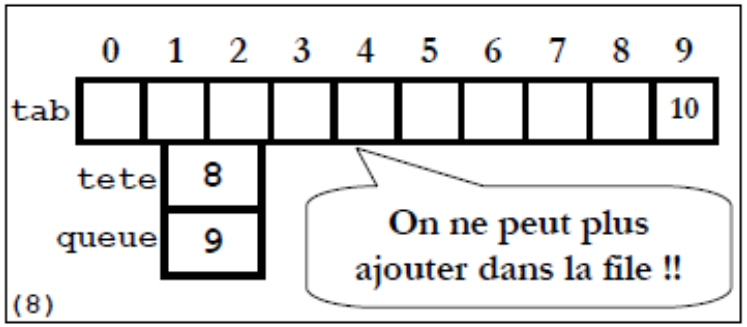


Représentation Contiguë d'une File (par tableau simple)









Spécification d'une file Contiguë

```
/* fichier "Tfile.h" */
#ifndef _FILE_TABLEAU
#define _FILE_TABLEAU
#include "Booleen.h"
// Définition du type file (implémentée par un tableau)
#define MAX_FILE 10 /* taille maximale d'une file */
typedef int Element; /* les éléments sont des int */
typedef struct {
 Element tab[MAX_FILE]; /* les éléments de la file */
 int tete; /* position précédant premier élément */
 int queue; /* position dernier élément */
} File;
// Déclaration des fonctions gérant la file
File file_vide ();
File enfiler (File f, Element e);
File defiler (File f);
Element tete (File f);
Booleen est_vide (File f);
#endif
```

Réalisation d'une file Contiguë (1)

```
/* fichier "Tfile.c" */
#include "Tfile.h"
// Définition des fonctions gérant la file
// initialiser une nouvelle file
File file_vide() {
 File f;
 f.queue = f.tete = -1;
 return f;
}
// tester si la file est vide
Booleen est_vide(File f) {
 if (f.tete == f.queue) return vrai;
 return faux;
}
// Valeur en tête de file
Element tete(File f) {
/* pré-condition : file non vide ! */
 if (est_vide(f)) {printf("Erreur: file vide !\n"); exit(-1);}
 return (f.tab)[f.tete+1];}
```

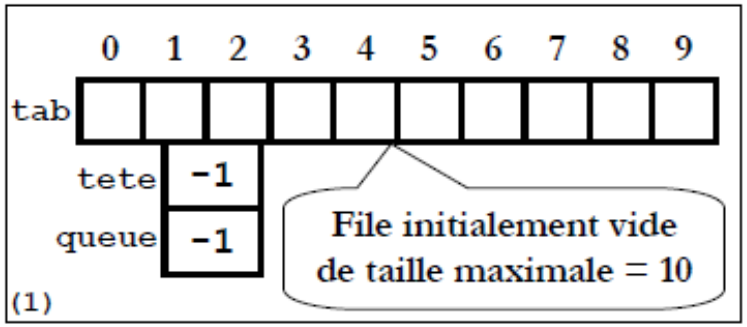
Réalisation d'une file Contiguë (2)

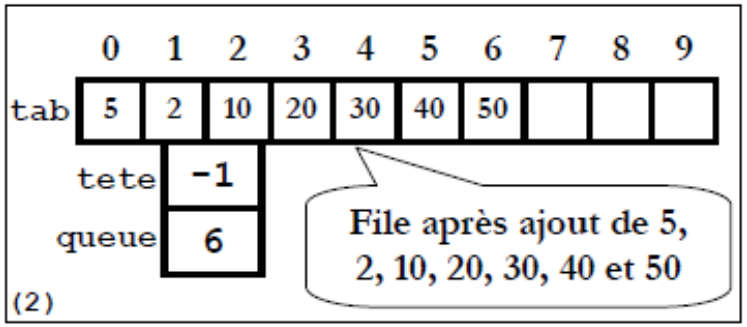
```
 // ajout d'un élément
File enfiler(File f, Element e) {
 if (f.queue == MAX_FILE-1) {
  printf("Erreur: on ne peut ajouter !\n");
  exit(-1);
 }
 (f.queue)++;
 (f.tab)[f.queue] = e;
 return f;
}
// enlever un élément
File defiler(File f) {
/* pré-condition : file non vide !*/
 if (est_vide(f)) {
  printf("Erreur: file vide !\n");
  exit(-1);}
 f.tete++;
 return f;
 }
```

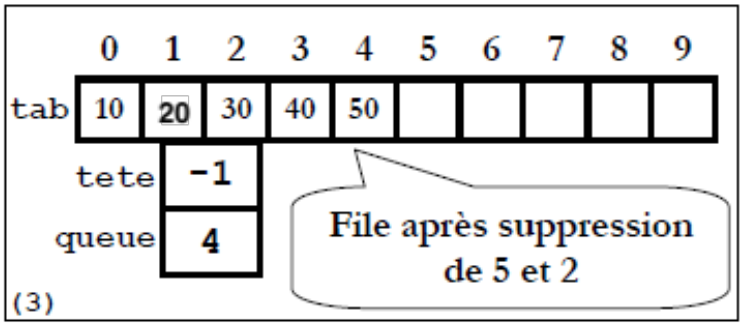
Représentation Contiguë d'une File (par tableau simple avec décalage)

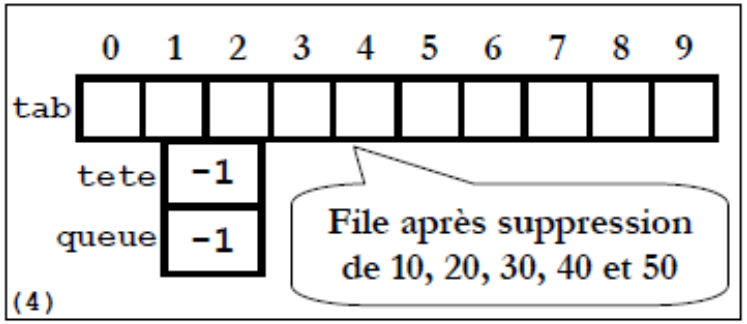
- Décaler les éléments de la file après chaque suppression
- Inconvénient : décalage très coûteux si la file contient beaucoup d'éléments

Représentation Contiguë d'une File (par tableau simple avec décalage)

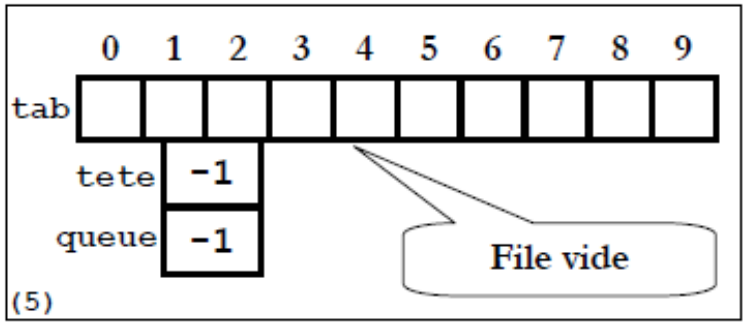


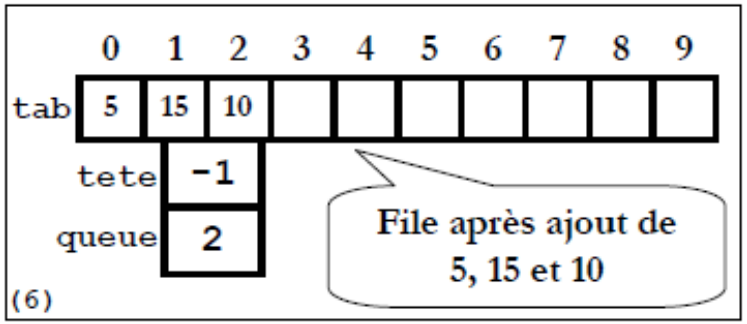


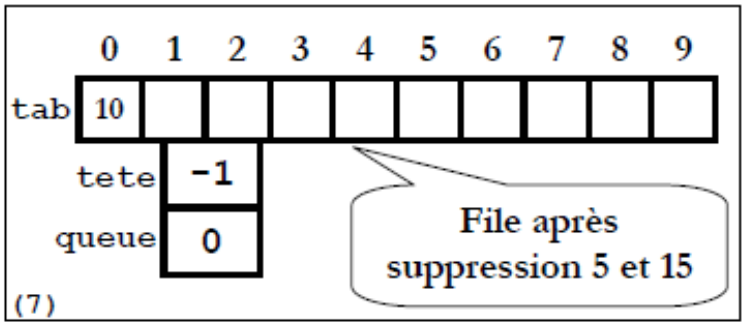


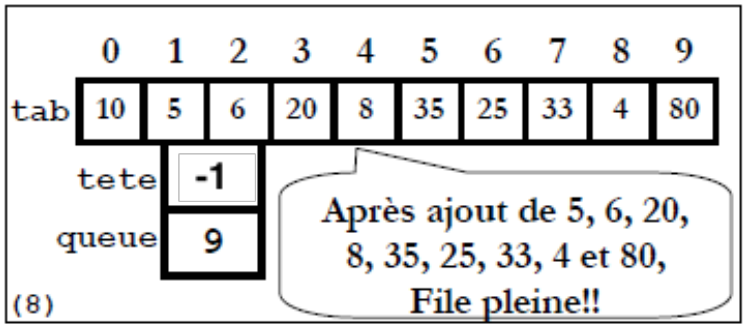


Représentation Contiguë d'une File (par tableau simple avec décalage)

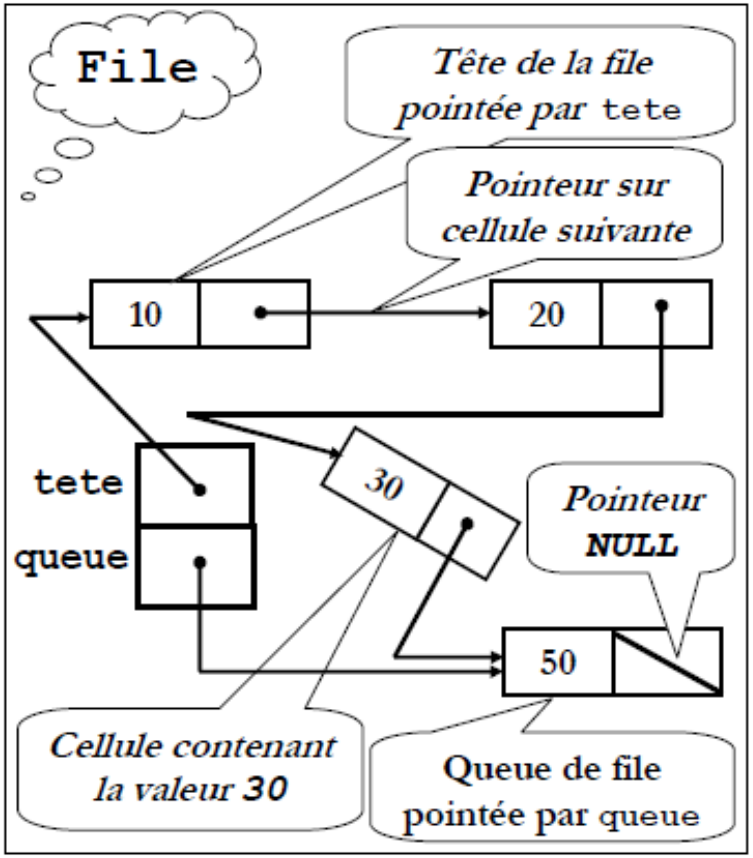








File chaînée



Spécification d'une File Chaînée

```
/* fichier "Cfile.h" */
#ifndef _FILE_CHAINEE
#define _FILE_CHAINEE
#include "Booleen.h"
typedef int Element; /* les éléments sont des int */
typedef struct cellule {
 Element valeur;
 struct cellule *suivant;
} Cellule;
typedef struct {
 struct cellule *tete;
 struct cellule *queue;
} File; // Définition du type File (implémentée par pointeurs)
File file_vide (); // Déclaration des fonctions
File enfiler (File f, Element e);
File defiler (File f);
Element tete (File f);
Booleen est_vide ( File f );
#endif
```

Réalisation d'une File Chaînée (1)

```
/* fichier "Cfile.c" */
#include "alloc.h"
#include "Cfile.h"
File file_vide() {
 File f;
 f.tete=f.queue=NULL;
 return f;
}
Booleen est_vide(File f) {
 return (f.tete==NULL)&&(f.queue==NULL);
}
Element tete(File f) {
/* pré-condition: file non vide ! */
 if (est_vide(f)) {
   printf("Erreur: file vide !\n");
   exit(-1);
 }
 return (f.tete)->valeur;
}
```

Réalisation d'une file Chaînée (2)

```
 File enfiler(File f, Element e) {
 Cellule * pc;
 pc=(Cellule *)malloc(sizeof(Cellule));
 pc->valeur=e;
 pc->suivant=NULL;
 if (est_vide(f) )
   f.tete=f.queue=pc;
 else f.queue=(f.queue)->suivant=pc;
 return f;
}
File defiler(File f) {
/* pré-condition: file non vide ! */
 if (est_vide(f)) {printf("Erreur: file vide !\n");exit(-1);}
 Cellule * pc;
 pc=f.tete;
 f.tete=(f.tete)->suivant;
 free(pc);
 if (f.tete==NULL) f.queue=NULL;
 return f;
}
```

Exemples d'application d'une file

- Gestion des travaux d'impression d'une imprimante :
  - Cas d'une imprimante en réseau, où les tâches d'impressions arrivent aléatoirement de n'importe quel ordinateur connecté. Les tâches sont placées dans une file d'attente, ce qui permet de les traiter selon leur ordre d'arrivée
- Ordonnanceur (dans les systèmes d'exploitation) :
  - Maintenir une file de processus en attente d'un temps machine ;
- Parcours « en largeur » d'un arbre (voir arbres)