UNIVERSITÉ PARIS SUD

Comprendre le monde, construire l'avenir®

TER Compilation

Par Rémi BUTET et Dorian GRANER

Introduction

- Création d'un compilateur :
  - Entrée : un fragment du langage C
  - Sortie : langage AMD64

- Travail de recherche : Allocateur de mémoire
  - Mémoire virtuelle et paging
  - Malloc et free

Sommaire

I. Conception du compilateur

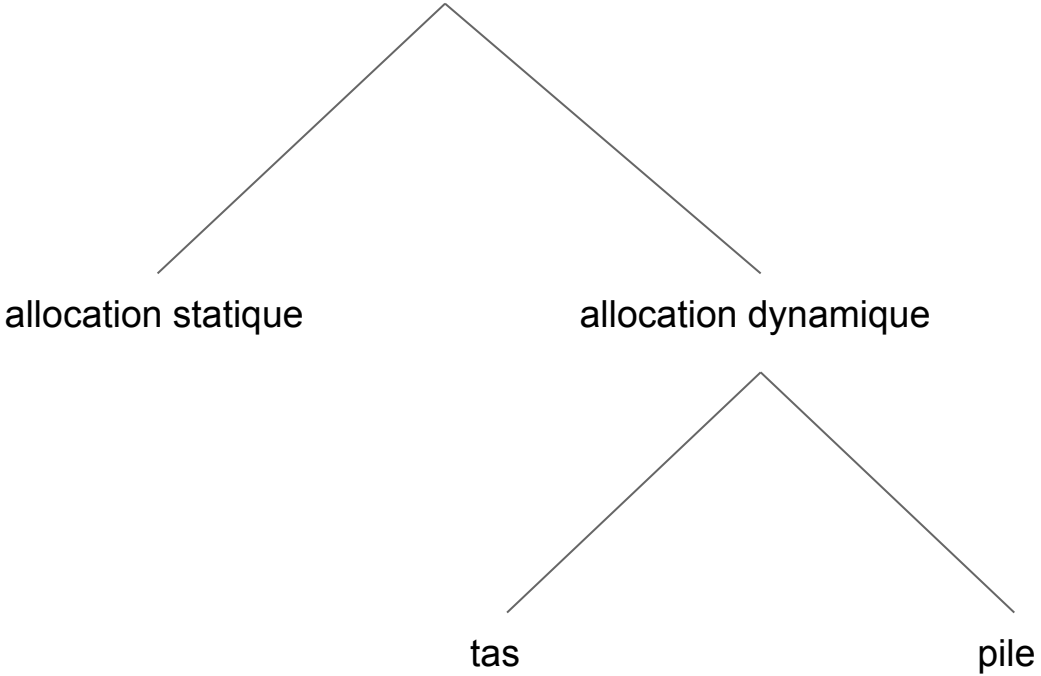
Exemple

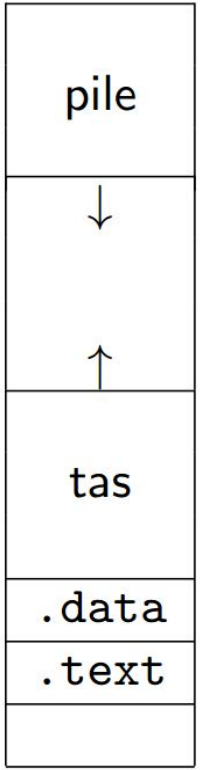
```
double* f;
extern int printf();
int fact(int n) {
  if (n <= 1) return 1;
  return n * fact(n - 1);
}

int main () {
    int i;
    i = fact(4);
    printf("i \x3A %d\n",i);
    printf("%d\n", &f);
    for(i=0; i<3; i++){
       f = f+i;
       printf("%d\n", f);
    }
    return 0;
}
```

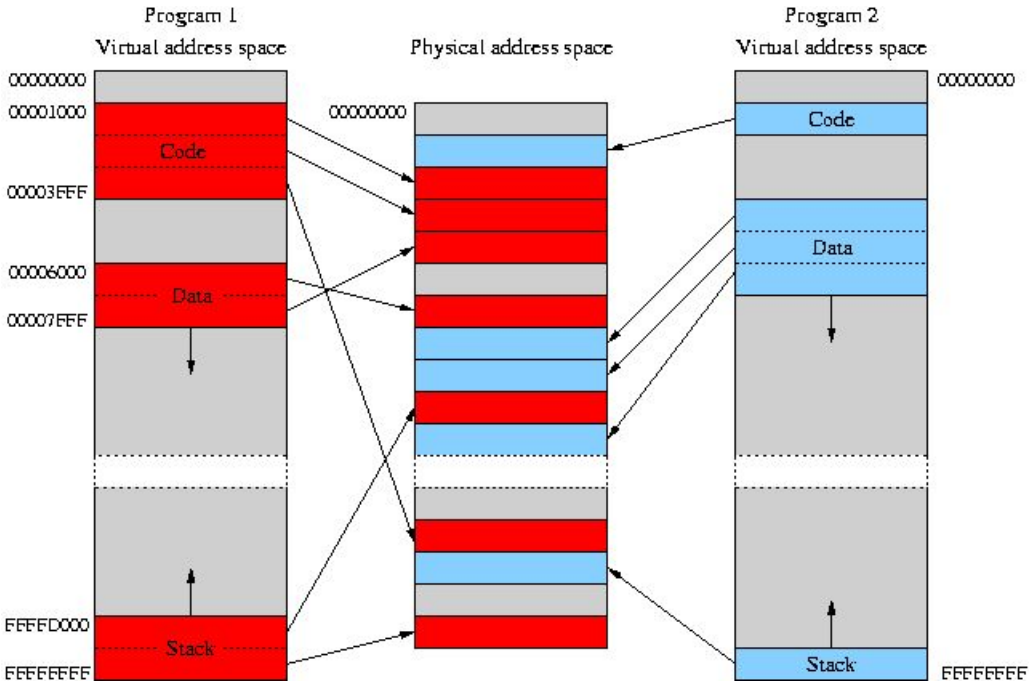
II. Allocateur de mémoire

Allocation mémoire

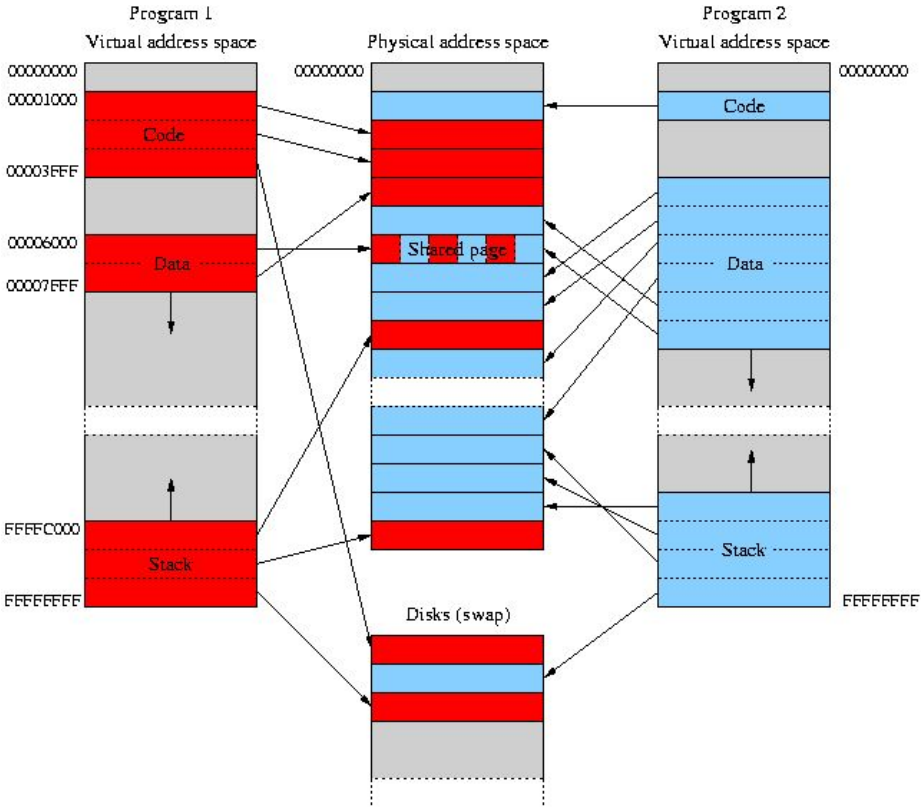




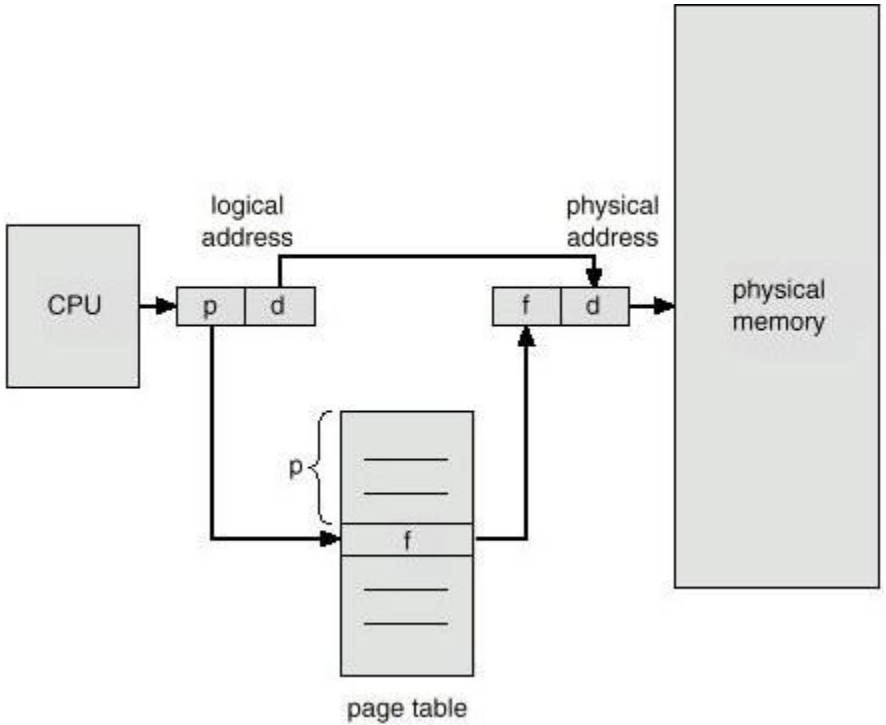
A.1. Mémoire virtuelle



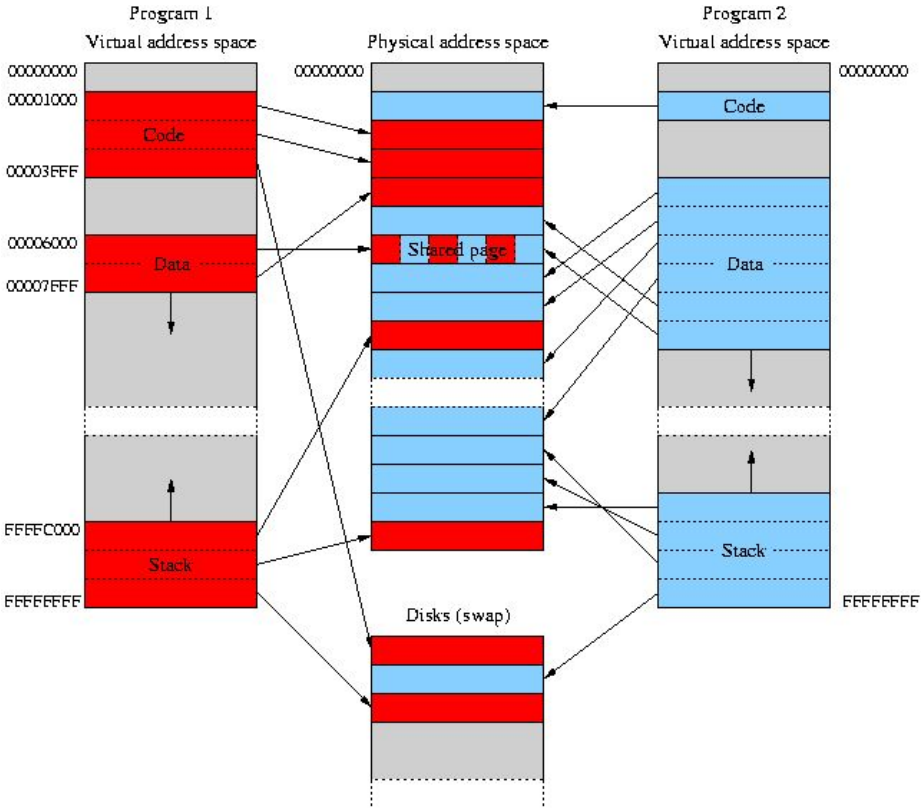
A.1. Mémoire virtuelle



A.2. Pagination



A.2. Pagination



B. Implémentation de malloc et free

```
void *malloc(int size);
```

La fonction ***malloc*** retourne un pointeur vers un nouveau bloc d'au moins ***size*** octets. Cependant en cas d'échec la fonction renverra NULL.

```
void free(void *ptr);
```

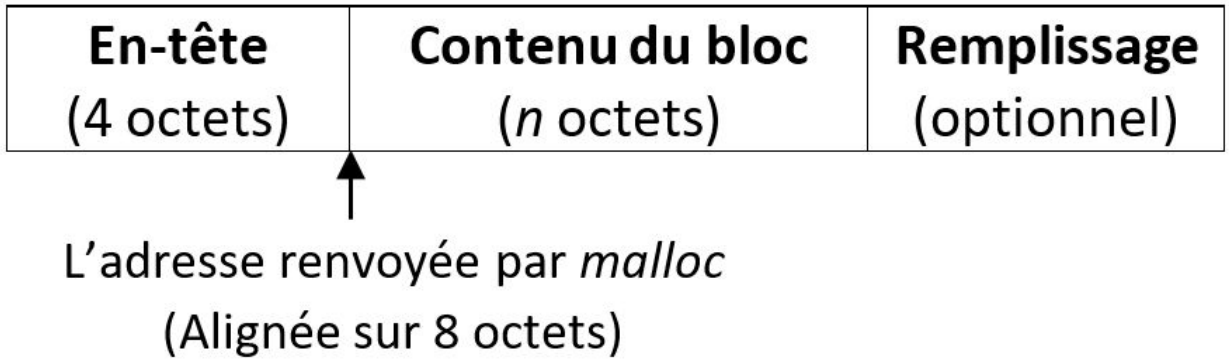
La fonction ***free*** prend comme argument l'adresse du premier octet du bloc alloué et ne retourne aucune valeur. Cependant si le pointeur n'a pas été alloué par la fonction malloc, il peut se produire une erreur.

B.1 Spécification de malloc et free

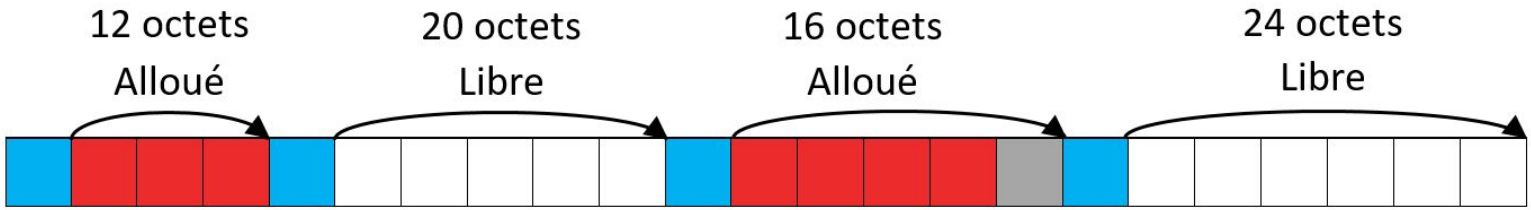
- Tous les blocs renvoyés par malloc doivent être alignés sur 8 octets
- Tous les blocs renvoyés par malloc ne peuvent plus être modifiés, ni déplacés
- Toutes les structures de données nécessaire à malloc et free doivent être stockées elle-même dans le tas.

B.2 Mise en œuvre

- Les blocs alloués ou libres sont tous **continus** en mémoire
- Liste **chaînée**

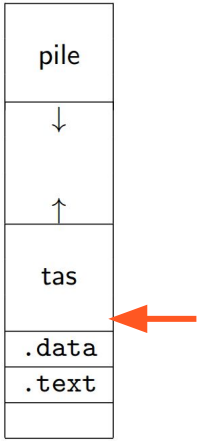


B.2 Mise en oeuvre : Exemple



- Légende : (un carré = 4 octets)

	En-tête
	Alloué
	Libre
	Remplissage



B.2 Mise en œuvre : En-tête

- La taille totale du bloc en octet
- Le statut : alloué ou libre

bit	5	4	3	2	1	0	Taille	Statut
...	0	1	0	0	0	1	Taille 16	Alloué
...	0	1	0	0	0	0	Taille 16	Libre
...	0	1	1	0	0	1	Taille 24	Alloué
...	1	0	0	0	0	0	Taille 32	Libre

B.3 Fonctionnement : malloc

malloc parcours toute la liste des blocs à la recherche d'un bloc libre suffisamment grand:

- Si il trouve un bloc :
  - Il le découpe éventuellement de la bonne taille (un alloué et un libre)
  - Il renvoie le pointeur du bloc alloué
- Sinon :
  - Sinon il alloue un nouveau bloc à la fin de la liste avec BRK
  - Il renvoie le pointeur du nouveau bloc alloué

B.3 Fonctionnement : free

free change uniquement le statut du bloc de alloué à libre

B.3 Fonctionnement : Stratégies

Pour trouver un bloc libre, il y a plusieurs stratégies possibles :

- First fit : Le premier bloc assez grand
- Next fit : Le premier bloc assez grand mais en commençant là où s'était arrêté la précédente recherche
- Best fit : Un bloc assez grand de taille minimale

B.4 Fusion des bloc libres : Problème

A force de découper les blocs, il y a de la **fragmentation** :

- Des blocs de la mémoire deviennent inutilisables
- Le temps de recherche augmente

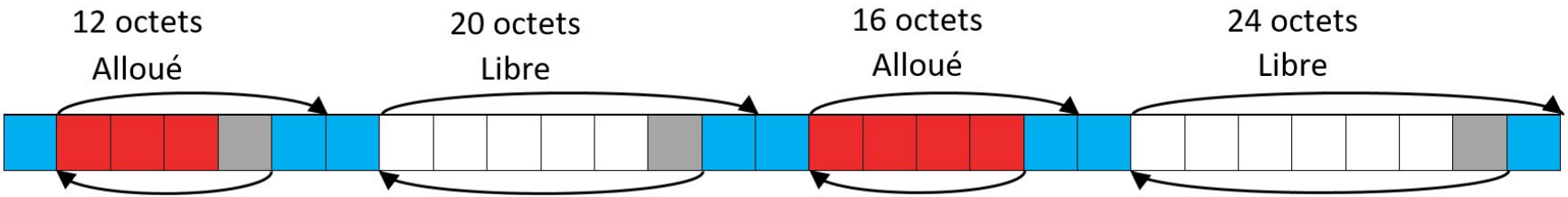
B.4 Fusion des bloc libres : Doublage

Pour éviter la fragmentation, il faut fusion les blocs libres adjacents :

- Facile : Fusionner avec le bloc **suivant** si il est libre
- Complexe : Fusionner avec le bloc **précédent** si il est libre

La solution est le **double chaînage** : dupliquer l'**en-tête** à la **fin** de chaque bloc

B.4 Fusion des bloc libres : Exemple



- Légende : (un carré = 4 octets)

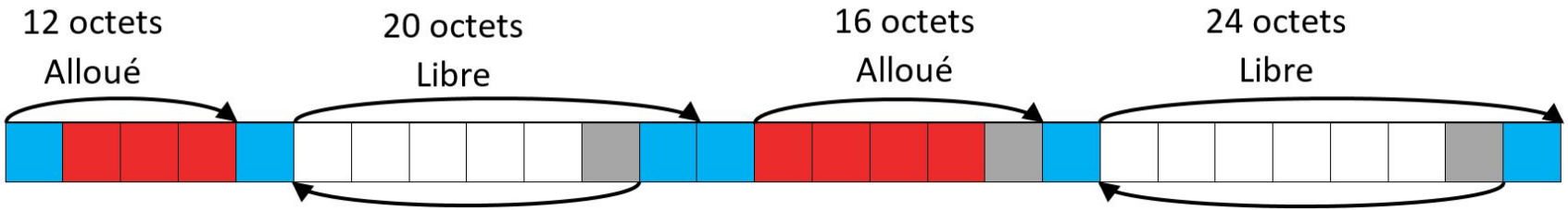
	En-tête
	Alloué
	Libre
	Remplissage

B.4 Fusion des bloc libres : Optimisation

Cette méthode est assez coûteuse en mémoire, mais il existe quelques améliorations qui permettent de réduire cela :

- Le faire uniquement pour les blocs libres
- Stocker dans l’entête du bloc le statut du bloc précédent

B.4 Fusion des bloc libres : Exemple



- Légende : (un carré = 4 octets)

	En-tête
	Alloué
	Libre
	Remplissage

bit	5	4	3	2	1	0	Taille	Statut	Précèdent
…	0	1	0	0	0	1	Taille 16	Alloué	Libre
…	0	1	0	0	0	0	Taille 16	Libre	Libre
…	0	1	1	0	1	1	Taille 24	Alloué	Alloué
…	1	0	0	0	1	0	Taille 32	Libre	Alloué

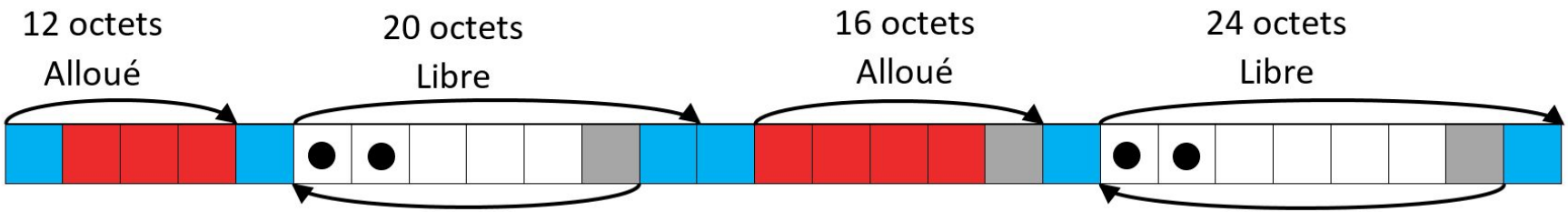
B.5 Free List

Parcourir toute la mémoire pour trouver des blocs libres assez grands reste quand même assez coûteux :

- Il est possible de **chaîner tous les blocs libres**
- Stocké les pointeurs dans les blocs libres

Inconvénient : Impose une taille de bloc minimale

B.5 Free List : Exemple



- Légende : (un carré = 4 octets)

	En-tête
	Alloué
	Libre
●	Libre (avec pointeur)
	Remplissage

B.5 Free List : Stratégies

Il y a désormais plusieurs stratégies possibles :

- Privilégié les blocs libres les plus anciens
- Privilégié les blocs libres les plus récent
- Trier les blocs libres par adresses croissantes
- Trier les blocs libres par taille
- Plusieurs listes de blocs libres organisées par taille
- ...

Conclusion

- Comprendre le fonctionnement d’un compilateur
- Approfondir nos connaissances
- Difficulté d’implémentation de ***malloc*** et ***free***