Allocation de la mémoire

Abdelkader Gouaïch

IUT de Montpellier

2018-2019

Allocation de la mémoire

- Allocation de la mémoire pour un processus : augmenter la taille du tas (heap)
- La limite de tas est appelée: program break (brk)
- En C l'allocation de la mémoire se fait en utilisant la famille de fonctions 'malloc'

- malloc utilise les fonctions de bas niveau brk() et sbrk()
- Pour le noyau, changer la taille du tas revient à changer la valeur du pointeur break
- Initialement, break est à la fin de la section des données non initialisées (eend)

- Quand on augmente break, le processus peut accéder à toutes les adresses dans la nouvelle zone
- Attention: les pages se sont pas effectivement allouées avec l'augmentation de break (mémoire virtuelle)
- Les pages seront allouées au premier accès réel

Signature de fonctions

int brk(**void**∗ ptr);
void∗ sbrk(intptr_t increment);

- brk() met le program break à la valeur indiquée par le pointeur (ptr)
- Comme la mémoire est allouée avec des pages (MV) la valeur du program break est arrondie à la fin de la page suivante

sbrk()

- sbrk modifie le progam break en incrémentant sa valeur actuelle de "increment"
- sbrk retourne en cas de succès la valeur précédente du program break
- Astuce: sbrk(0) retourne la valeur actuelle du programme break.

malloc et free

void∗ malloc(size_t size);

- malloc retourne le pointeur sur un bloc mémoire alloué
- attention la taille du bloc peut être supérieure à celle demandée
- arrondie pour avoir un bloc multiple de 8 ou 16

free

void free(**void**∗ ptr)

- free libère le bloc pointé par ptr
- Attention : ptr doit être un pointeur donné par malloc
- free ne modifie pas le program break !
- Elle met simplement le bloc dans une liste de recyclage pour le prochain malloc

- Autres fonction d'allocation sur la tas
- void* calloc(size_t n, size_t size);
- Allocation de n*size blocs de mémoire
- initialisé à 0 (NULL)

realloc

void* realloc(**void*** ptr, size_t size);

- Modification de la taille du bloc déjà alloué
- ptr est l'ancien bloc à modifier
- size nouvelle taille
- retourne le nouveau pointeur

Allocation dans la section stack

- Il est possible d'allouer dynamiquement la mémoire dans la pile !
- Ceci est possible car la fonction qui fait l'allocation est toujours sur le top de la frame stack (pile)
- Nous pouvons alors modifier la taille de la pile en modifiant la frame de la fonction courante

void* alloca(size_t size);

- Allocation d'un bloc mémoire de taille size sur la pile.

- Pas besoin de free !
- Au retour la frame disparait et avec les blocs alloués dynamiquement

- Avantages de alloca par rapport à malloc:
- rapidité
- pas de free
- facilite la gestion de la mémoire en cas de sauts (longjmp)
- Mais attention:
- Le pointeur n'a de sens que pour l'appel de la fonction
- Le pointeur n'a plus de sens dès qu'il y a un return