Linked-list Presentation

ACU 2025 Team

Copyright

Rules

- You must have downloaded your copy from the Assistants' Intranet <https://intra.forge.epita.fr>.
- This document is strictly personal and must **not** be passed on to someone else.
- Non-compliance with these rules can lead to severe sanctions.

Linked-list implementation

Variant - First fit

- Simplest to implement
- Fast execution

Management

Allocation

- User does a malloc of size s.
- Traverse the list and find the first block with a size of at least s.
- If the block is larger than necessary, the block is split in two:
  - One with the necessary space
  - Another with the rest of the space.

Free

- Mark the block as free.
- To limit fragmentation, you can merge neighbor blocks that are free (fusion operation).

Metadata

At the beginning of each block, you will need the following metadata:

- The block status (allocated/free)
- The block size
- The pointer to the next block
- The pointer to the previous block (if you intend to merge neighbor blocks when freeing)

Example

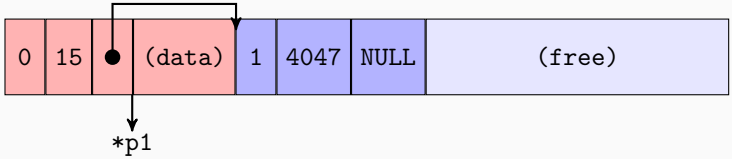
Step 1

```
void *p1 = malloc(15);
```

- You have to allocate the first page with `mmap`.
- Here is the first free block which takes up the whole page.

Step 2

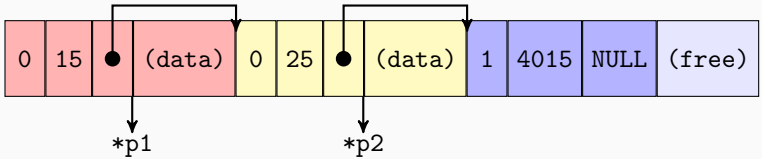
```
void *p1 = malloc(15);
```



- We want to allocate a block of size 15.
- We find the first block that matches the size requirement (which is also the only block in the page).
- We split the block in two in order to limit wasting memory.

Step 3

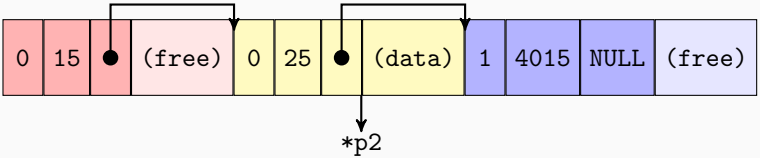
```
void *p1 = malloc(15);
void *p2 = malloc(25);
```



- We want to allocate another block of size 25.
- Same process as before.

Step 4

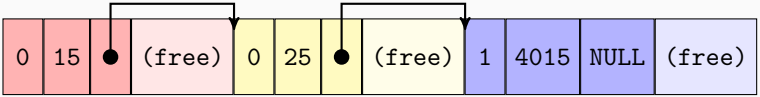
```
void *p1 = malloc(15);
void *p2 = malloc(25);
free(p1);
```



- We free the first block.
- We encounter fragmentation.

Step 5

```
void *p1 = malloc(15);
void *p2 = malloc(25);
free(p1);
free(p2);
```



- We free the second block.
- The neighbors blocks are free so we can merge them (fusion operation).

Step 6

```
void *p1 = malloc(15);
void *p2 = malloc(25);
free(p1);
free(p2);
```

- In order to do that, we need a pointer to the previous block.
- The whole page is now free.

Improvements

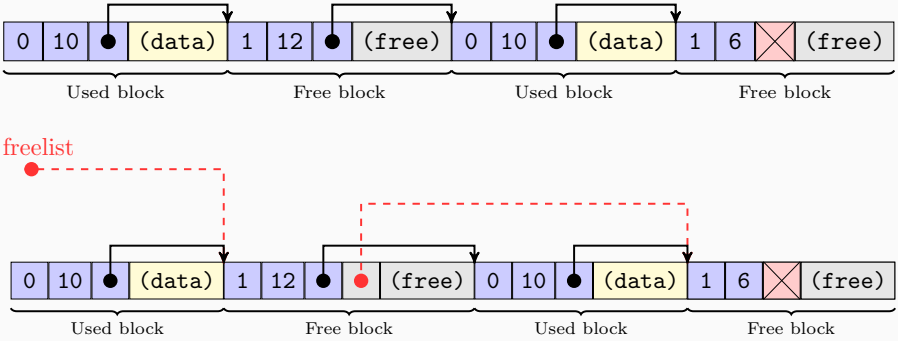
Cons of first-fit

- Heavy fragmentation, accumulation of small blocks.
- Allocation is fast but space is wasted even with split and fusion operations.
- Variant: Best-fit

Variant - Best-fit

- Traverse the whole list to find the block that fits the size requirement the best.
- Memory efficient.
- Checking the whole memory is slow.

Example using free-list



- Allocation is faster since we only search through free blocks.

Conclusion

Conclusion

- Simple implementation.
- Fast allocation but high fragmentation.
- Not memory efficient.