Operating Systems Design

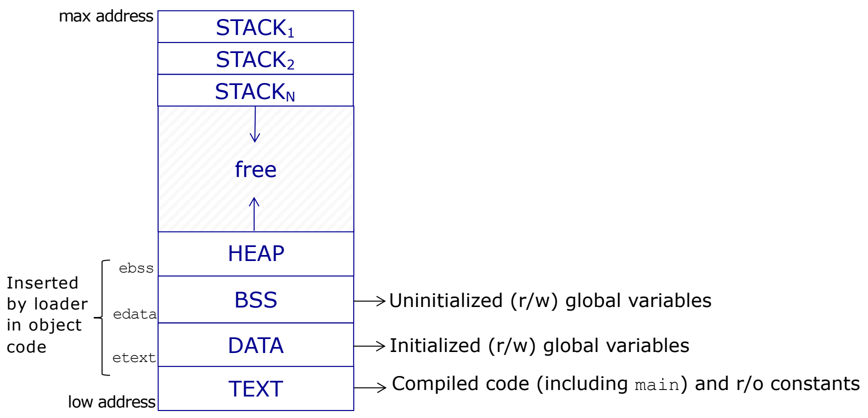
Xinu Memory Management

Instructor: Michela Becchi

1

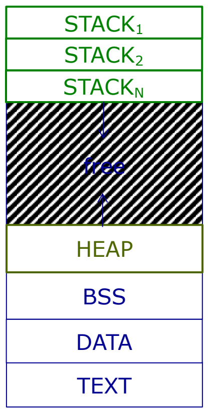
Memory management in Xinu

- A single address space shared by all processes
  - Each process has its own stack
  - Xinu processes are "threads" (lightweight processes)


max address
STACK$_1$
STACK$_2$
STACK$_N$
free
HEAP
ebss
BSS
edata
DATA
etext
TEXT
low address
Inserted by loader in object code
Uninitialized (r/w) global variables
Initialized (r/w) global variables
Compiled code (including main) and r/o constants

2

Dynamic memory allocation in Xinu



`meminit` – initializes free list

`getstk` – called by `create`
`freestk` – called by `kill`

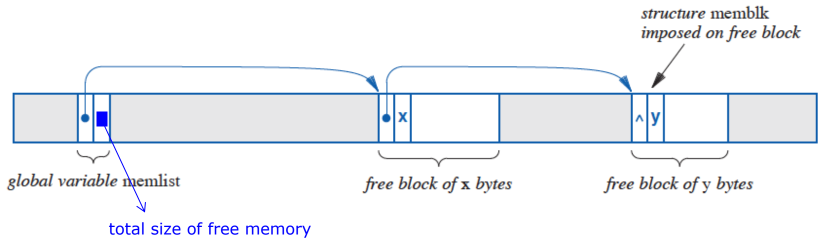
`getmem` – the system does not keep track of which process allocated the heap space

`freemem` – must explicitly be invoked: the system does not automatically release heap space

FRAGMENTATION!

Free list in Xinu

- Linked list of free blocks ordered by increasing address
- Stored in free space
- `memlist` = pointer to first free block
- Each block (`memblk`) contains:
  - Pointer to next block
  - Size of the block (except for `memlist`)
- Initialized in `meminit.c`



Xinu data structures for memory management

- In memory.h

```
/* Block of free list */
struct  memblk  {
        struct  memblk  *mnext;         /* Ptr to next free memory blk        */
        uint32  mlength;                /* Size of blk (includes memblk header)*/
        };

extern  struct  memblk  memlist;        /* Head of free memory list     */

extern  void    *minheap;               /* Start of heap                */
extern  void    *maxheap;               /* Highest valid heap address   */


/* Added by linker */
extern  int     text;                   /* Start of text segment        */
extern  int     etext;                  /* End of text segment          */
extern  int     data;                   /* Start of data segment        */
extern  int     edata;                  /* End of data segment          */
extern  int     bss;                    /* Start of bss segment         */
extern  int     ebss;                   /* End of bss segment           */
```

Xinu – memory requests rounding

- memblk must contain at least 8 bytes
- memory allocation requests are rounded to multiple of memblk size (8 bytes)
- see in memory.h
  - roundmb(x)
  - truncmb(x) – only used at startup on initial free block size – see meminit.c

Xinu - heap space allocation & release

- `getmem`
  - uses first-fit allocation policy
  - splits the block if necessary
- `freemem`
  - uses address to locate block in free list
  - tries to coalesce (to limit fragmentation)
    - with previous free block, next free block, or both

Xinu – stack space allocation & release

- `getstk`
  - allocates stack from highest block in free list that fits the request
  - visits whole free list to find suitable block
  - splits block if necessary
  - returns the *highest* address in the block
- `freestk`
  - uses `freemem`
  - converts the address to be passed to the `freemem` from highest address in the block (returned by `getstk` and passed as its argument) to lowest address in the block