MP2-Implementation

r09922136 廖婕吟

1. VMA management in process's user address space

1) VMA structure is defined in `kernel/proc.h`

```
// Virtual Memory area
struct vma {
  struct spinlock lock;

  // p->lock must be held when using these:
  uint64 start;       // VMA start, inclusive
  uint64 end;         // VMA end , exclusive
  struct vma *next;   // list of VMA's
  uint64 length;      // 0 means vma not used
  uint64 off;         // offset within file
  int page_prot;      // access permissions
  int flags;          // shared or private
  struct file *file;  //mapped file, if any
};
```

A fixed-size array `vma_list[]` in length 16 of VMAs is declaired in `kernel/proc.c`, and we allocate VMA from this array as needed.

2) VMAs is allocate from the middle of the process's user address space since defining `VMA_START` at the position of `MAXVA / 2`, and the direction we allocate the VMA is from the lower address to the higher address.

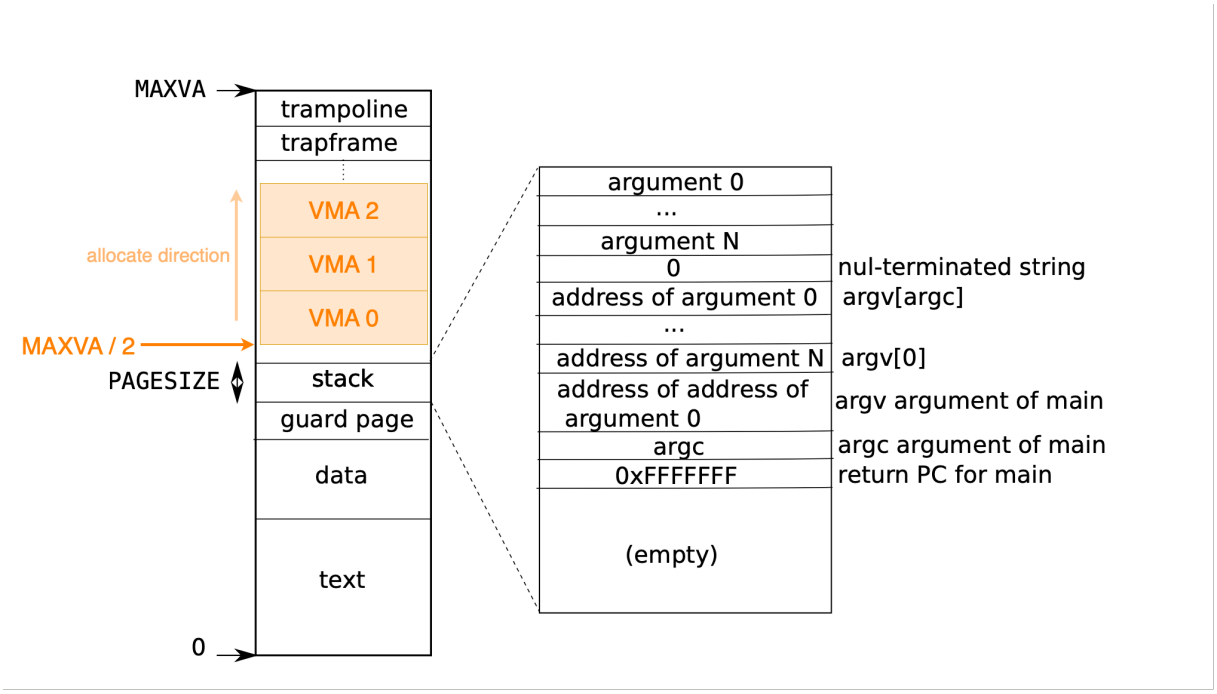

Figure 1: Allocate VMAs from the middle of a process's user address space

3) The address that we start to allocate a VMA (`uint64 start`) which is declare in the `struct vma` is always round to a multiple of the page size. that is we allocate VMA with page-aligned.

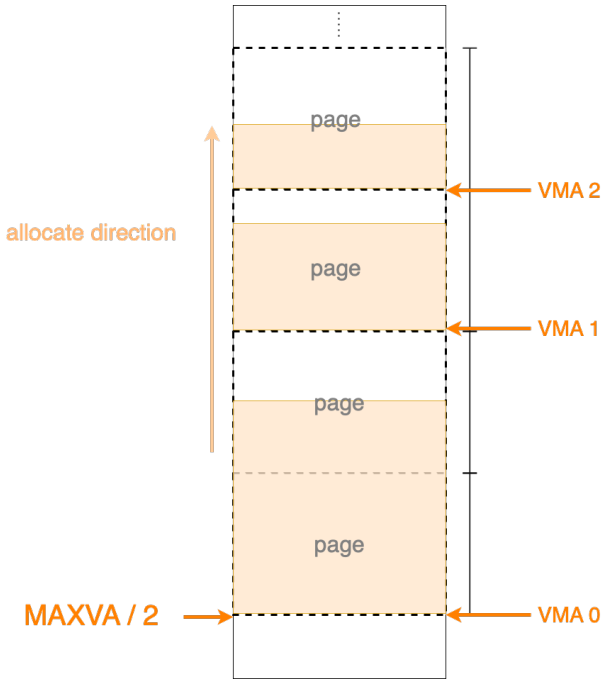

Figure 2: Allocate VMA with page aligned

4) The memory area that we suppose to `sys_munmap()` is only can be unmap from the start or from the end of a VMA, which means we don't support splitting a VMA into two VMAs when unmapping.

5) Use link list data structure to maintain the **process's table of mapped regions**. When `sys_mmap()` is called, the new VMA will be added into the list of VMAs by using the vma pointer (`struct vma *next`) which declair in the `struct vma`.

2. Assumptions

1) `addr` will always be zero.
2) `offset` will always be zero, that is, we always map the file from the starting point of the mmap-ed file.
3) When `length` is 0, the VMA is not used.
4) `flags` will be one of the following bits
   - `MAP_SHARED`: modifications to the mapped memory should be written back to the file.
   - `MAP_PRIVATE`: modifications should not be written back.
5) The VMA is page-aligned when we `sys_mmap()`.
6) the `sys_munmap()` is not supported to unmap the area that is at the middle of a VMA, that is only can munmap from the start or from the end of a VMA and won't split a VMA into two VMAs.