بسم الله الرحمن الرحیم

پروژه پنجم درس آزمایشگاه سیستم عامل

دکتر کارگهی

محمد امین توانایی - ۸۱۰۱۰۱۳۹۶

سید علی تهامی - ۸۱۰۱۰۱۳۹۷

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

سوال ۱

در لینوکس ناحیه مجازی روشی که که در آن هر پردازه یک فضای آدرس دهی مجزا داده می شود حتی اگر حافظه فیزیکی به صورت پیوسته نباشد این موضوع به صورت مجازی فراهم می شود. این سیستم شامل دو بخش فضای آدرس دهی مجازی و نگاشت حافظه است. اما در xv6 به صورت بسیار ساده ای پیاده شده است که فضای از همان ابتدا الوکیت می شود و آدرس دهی و صفحه بندی و ایزوله کردن بسیار ساده پیاده سازی شده.

سوال ۲

در این روش ما می توانیم جدول را به چندین جدول بشکنیم و با این کار می توانیم قسمت های جدول را به صورت مجزا برگزاری کنیم و همچنین در مواقعی که جدول در یک صفحه جا نمی شود این روش می تواند با تقسیم آن این مشکل را برطرف کند. در این روش ابتدا در جدولی مشخص می شود که آدرس مربوطه در کدام زیر جدول است و سپس از زیر جدول آدرس مربوطه استخراج می شود.

سوال ۳

آدرسی که برنامه می دهد ۳ بخش است ۱۰ بیت اول آدرس مربوط به جدول مدخل هاست و ۱۰ بیت دوم برای آدرس در جدول صفحه دومی و در نهایت ۱۲ بیت هم آفست برای همان صفحه است که مستقیم اعمال می شود. ردیف های جدول مدخل با ۲۰ بیت جدول سطح ۲ را مشخص می کنند و ۱۲ بیت هم مربوط به پرچم هاست که موارد مختلفی مانند سطح دسترسی را می تواند مشخص کند همچنین در جدول ثانوی هم ردیف ها مثل جدول اصلی است.

سوال ۴

```
//PAGEBREAK: 21
// Free the page of physical memory pointed at by v,
// which normally should have been returned by a
// call to kalloc().  (The exception is when
// initializing the allocator; see kinit above.)
```

طبق چیزی که در کد نوشته آدرس دهی به صورت فیزیکی است.

سوال ۵

```
// Create PTEs for virtual addresses starting at va that refer to
```

```
// physical addresses starting at pa. va and size might not
// be page-aligned.
```

برای اختصاص فضای حافظه مجازی استفاده می شود.

سوال ۷

```
// Return the address of the PTE in page table pgdir
// that corresponds to virtual address va.  If alloc!=0,
// create any required page table pages.
static pte_t *
walkpgdir(pde_t *pgdir, const void *va, int alloc)
{
  pde_t *pde;
  pte_t *pgtab;

  pde = &pgdir[PDX(va)];
  if(*pde & PTE_P){
    pgtab = (pte_t*)P2V(PTE_ADDR(*pde));
  } else {
    if(!alloc || (pgtab = (pte_t*)kalloc()) == 0)
      return 0;
    // Make sure all those PTE_P bits are zero.
    memset(pgtab, 0, PGSIZE);
    // The permissions here are overly generous, but they can
    // be further restricted by the permissions in the page table
    // entries, if necessary.
    *pde = V2P(pgtab) | PTE_P | PTE_W | PTE_U;
  }
  return &pgtab[PTX(va)];
}
```

این تابع در واقع آدرس مجازی را گرفته و آدرس فیزیکی را خروجی میدهد

به این صورت که :

اول page directory موردنظر را پیدا کرده و با استفاده از آن آدرس فیزیکی page table را پیدا میکند

اگر page table موجود نباشد و allocate یک باشد تخصیص حافظه انجام میشود

در آخر خانه ای از آن page table که va مشخص شده است که آدرس فیزیکی مربوطه است را خروجی میدهد.

سوال ۸

```
// Create PTEs for virtual addresses starting at va that refer to
// physical addresses starting at pa. va and size might not
// be page-aligned.
static int
mappages(pde_t *pgdir, void *va, uint size, uint pa, int perm)
{
  char *a, *last;
  pte_t *pte;

  a = (char*)PGROUNDDOWN((uint)va);
  last = (char*)PGROUNDDOWN(((uint)va) + size - 1);
  for(;;){
    if((pte = walkpgdir(pgdir, a, 1)) == 0)
      return -1;
    if(*pte & PTE_P)
      panic("remap");
    *pte = pa | perm | PTE_P;
    if(a == last)
      break;
    a += PGSIZE;
    pa += PGSIZE;
  }
  return 0;
}
```

این تابع برای نگاشت ورودی های یک page table به آدرس های فیزیکی استفاده می شود و به این صورت عمل میکند که page directory فعلی را به عنوان آرگومان می گیرد و سپس از آدرس مجازی فعلی که در حال پردازش است تا last که آخرین آدرسی که باید نگاشت کند را نگاشت میکند.

```
// Allocate page tables and physical memory to grow process from oldsz
to
// newsz, which need not be page aligned.  Returns new size or 0 on
error.
int
allocuvm(pde_t *pgdir, uint oldsz, uint newsz)
{
  char *mem;
  uint a;

  if(newsz >= KERNBASE)
    return 0;
```

```
  if(newsz < oldsz)
    return oldsz;

  a = PGROUNDUP(oldsz);
  for(; a < newsz; a += PGSIZE){
    mem = kalloc();
    if(mem == 0){
      cprintf("allocuvm out of memory\n");
      deallocuvm(pgdir, newsz, oldsz);
      return 0;
    }
    memset(mem, 0, PGSIZE);
    if(mappages(pgdir, (char*)a, PGSIZE, V2P(mem), PTE_W|PTE_U) < 0){
      cprintf("allocuvm out of memory (2)\n");
      deallocuvm(pgdir, newsz, oldsz);
      kfree(mem);
      return 0;
    }
  }
  return newsz;
}
```

این تابع با تخصیص حافظه های مجازی به فیزیکی فضای آدرس کاربر را افزایش میدهد و به این صورت عمل میکند که :
چک میکند اگر فضای آدرس نیاز به بزرگتر شدن دارد یا نه همچنین چک میکند که افزایش آدرس آیا به حافظه کرنل تجاوز میکند یا خیر در اگر مثبت باشد تخصیص صورت نمیگیرد.
و پس از آن حافظه های قبلی را آزاد کرده و حافظه های جدید را با استفاده از تابع mappages تخصیص صورت خواهد گرفت.

سوال ۹

مراحل بارگذاری یک برنامه به حافظه زمانی که سیستم کال exec در xv6 فراخوانی می‌شود به شرح زیر است:

I. **حذف وضعیت حافظه**: ابتدا، سیستم کال exec وضعیت حافظه فرآیند فراخوانی را پاک می‌کند.
II. **یافتن فایل برنامه**: سپس به سیستم فایل مراجعه کرده و فایل برنامه مورد نظر را پیدا می‌کند.
III. **اختصاص جدول صفحه جدید**: در این مرحله، یک جدول صفحه جدید بدون نگاشت‌های کاربری با استفاده از `setupkvm()` ایجاد می‌شود.
IV. **اختصاص حافظه برای هر بخش ELF**: حافظه برای هر بخش ELF با استفاده از `allocuvm()`

اختصاص داده می‌شود. این تابع وظیفه افزایش حافظه مجازی کاربر در یک دایرکتوری صفحه مشخص را بر عهده دارد. این کار از طریق تخصیص صفحات جدید حافظه فیزیکی و نگاشت آن‌ها به فضای آدرس مجازی فرآیند انجام می‌شود.

V. **بارگذاری هر بخش به حافظه**: هر بخش از فایل ELF با استفاده از `loaduvm()` به حافظه بارگذاری می‌شود. تابع `loaduvm()` از `walkpgdir` برای یافتن آدرس فیزیکی حافظه اختصاص داده‌شده استفاده کرده و داده‌های هر صفحه از بخش ELF را با استفاده از `readi` از فایل می‌خواند.

VI. **تنظیم وضعیت رجیسترها**: در این مرحله، وضعیت رجیسترها از جمله PC و EIP به ورودی برنامه و ESP به بالای پشته کاربر مقداردهی اولیه می‌شود.

VII. **به‌روزرسانی جدول صفحه**: در پایان، با استفاده از `switchuvm()` جدول صفحه در سخت‌افزار به‌روزرسانی می‌شود و سپس با `freevm()` جدول صفحه قبلی از حافظه حذف می‌گردد.

کد آزمون

نتیجه بدون استفاده و با استفاده از قفل: (سمت چپ بدون قفل)