فهرست

۱ دستورکار شماره سه:

- پیاده‌سازی سازوکارهای لازم برای بارگذاری و اجرای برنامه‌های کاربردی، مدیریت فراخوانی‌های سیستمی و ایجاد محیطی امن برای اجرای برنامه‌ها در فضای کاربر.

۲ اهداف آموزشی:

- آشنایی با نحوه اجرای برنامه‌های کاربر در Pintos
- پیاده‌سازی پشته راه‌اندازی فرآیند (Argument Passing)
- مدیریت ایمن حافظه کاربر و جلوگیری از Kernel Panic
- پیاده‌سازی فراخوانی‌های سیستم (System Calls)
- مدیریت فایل‌ها، descriptorها و همگام‌سازی پردازه‌ها
- تکمیل زنجیره exit → run → load → exec.

۳ توضیح پروژه:

- در پروژهٔ دوم Pintos، هدف ایجاد بستری است که برنامه‌های کاربر بتوانند در محیط Pintos اجرا شوند. در این پروژه باید:
  - پردازه جدید ایجاد کنید (exec)،
  - منتظر پایان آن بمانید (wait)،
  - حافظه کاربر را بررسی کنید،
  - فراخوانی‌های سیستم را پیاده‌سازی کنید،
  - و تعامل با فایل‌سیستم را فعال کنید.

۳-۱ ساختار کلی پروژه Pintos:

- مسیر پروژه: /pintos/src/userprog
- فایل‌های مهم:
  - process.c — بارگذاری ELF و ایجاد stack
  - syscall.c — مدیریت فراخوانی‌های سیستمی
  - exception.c — مدیریت خطاها و صفحه‌بندی (page fault)
- تست‌ها در مسیر: /pintos/src/tests/userprog.

۳-۲ مراحل انجام آزمایش:

۲-۳-۱ فاز اول – شناخت ساختار و آماده‌سازی

- اجرای اولیه Pintos بدون هیچ تغییری
- مرور فایل‌های process.c, syscall.c, exception.c
- بررسی عملکرد load و start_process با gdb
- اجرای اولیه برنامه‌های نمونه (مثلاً echo) بدون argument

۲-۳-۲ فاز دوم – طراحی:

- انتخاب روش مدیریت حافظه کاربر
- طراحی جدول توصیف فایل‌ها (File Descriptor Table)
- طراحی همگام‌سازی بین والد و فرزند برای exec و wait
- طراحی ساختار نگهداری فرایندهای فرزند (child processes)

۲-۳-۳ فاز سوم – Argument Passing:

- پیاده‌سازی تابع setup_stack
- قراردادن رشته‌های argv در بالای پشته
- آرایهٔ آدرس‌ها، مقدار argc و fake return
- تست با:
  - 'pintos -- run 'echo a b c
- اطمینان از قرار گرفتن صحیح argv[i] در برنامه کاربر

۲-۳-۴ فاز چهارم – دسترسی امن به حافظه کاربر:

- جلوگیری از دسترسی مستقیم کرنل به pointerهای کاربر
- بررسی آدرس‌ها با pagedir_get_page
- مدیریت نقض صفحه مناسب در page_fault
- رفتار صحیح هنگام ورودی نامعتبر به system call
- تست: pointer = NULL، pointer خارج از USER_VADDR

۲-۳-۵ فاز پنجم – زیرساخت System Call:

- تکمیل تابع syscall_handler
- استخراج شماره system call از esp کاربر
- انتقال آرگومان‌ها از پشته
- dispatch فراخوانی‌ها با switch یا جدول توابع
- مقدار برگشتی در EAX نوشته شود

۲-۳-۶ فاز ششم – پیاده‌سازی فراخوانی‌های فایل و پردازه:

الف) فراخوانی‌های پردازه

- halt()
- exit(status)
- چاپ پیام خروج
- exec(cmd_line)
- همگام‌سازی load بین والد و فرزند
- wait(pid)

ب) فراخوانی‌های فایل

- remove()، create()
- Open()
- Filesize()
- read()، write()
- رفتار خاص stdin و stdout
- tell()، seek()
- close()
- نکات مهم
  - جلوگیری از write روی فایل اجرایی در حال اجرا
  - مدیریت همزمانی دسترسی به فایل‌سیستم
  - آزادسازی منابع هنگام exit

۲-۳-۷ فاز هفتم – ارزیابی و تست

- اجرای تست‌ها:
  - make check
- بررسی خروجی تست‌ها در مسیر: tests/userprog/
- تست موفقیت‌آمیز بودن تمام موارد زیر:
  - Argument passing، bad ptr، file syscalls، exec/wait، write/read/close
- رفع خطاها و panicهای احتمالی
- جمع‌بندی و مستندسازی نهایی

۳-۳ سؤالات تحلیلی پایانی

۱. چرا برای exec لازم است والد منتظر باقی بماند تا نتیجه بارگذاری فرزند مشخص شود؟
۲. ساختار داده‌ای شما برای مدیریت فرزندان یک فرآیند چگونه است؟
۳. مدیریتpointer های نامعتبر چگونه انجام می‌شود و چرا؟
۴. چرا write روی فایل اجرایی باید ممنوع شود؟
۵. هنگام exit یک فرآیند، دقیقاً چه منابعی باید آزاد شوند؟
۶. تفاوت write روی stdout با write روی فایل چیست؟
۷. در صورت ارسال pointer نامعتبر به read یا write، رفتار صحیح چیست؟
۸. طراحی File Descriptor Table شما چگونه است و چرا این مدل را انتخاب کردید؟
۹. در ساخت پشته برنامه کاربر برای argv چگونه ترتیب و هم‌ترازی رعایت شد؟

۱۰. اگر دو فرآیند هم‌زمان به یک فایل write کنند، چه اتفاقی می‌افتد و چگونه همگام‌سازی کردید؟

۴ بخش امتیازی:

- بر اساس نسخه لینوکس که در اختیار دارید، یک System Call به کرنل سیستم عامل خود اضافه کنید.
- عمل این تابع می‌تواند اضافه کردن پیغامی در Log سیستم یا انجام اعمال ساده ریاضی شبیه به جمع باشد.
- گزارش باید شامل توضیحاتی درباره System Call و نحوه اضافه کردن یک System Call جدید به کرنل، نتیجه اجرا و توضیح کد برنامه باشد.

۵ تحویل پروژه:

- گزارشکار به همراه سورس کدهای خود را پوشه‌ای با نام osLab_P1_stdID ارسال کنید.
- مهلت ارسال یکشنبه ۱۴۰۴/۰۹/۰۹ ساعت ۱۳ می‌باشد.
- ارائه حضوری یکشنبه ۱۴۰۴/۰۹/۰۹ در زمان برگزاری کلاس خواهد بود.

موفق باشید - آهوز