به نام خدا

دانشکده مهندسی کامپیوتر

# توضیحات پروژه
# Container runtime system

درس: سیستم‌های عامل
نیم‌سال دوم ۰۴-۰۳
استاد: دکتر جلیلی

## مقدمه

در دنیای مدرن، محفظه[1]ها به‌عنوان یکی از اساسی‌ترین فناوری‌ها در دنیای نرم‌افزار و سیستم‌عامل‌ها به شمار می‌روند. سیستم‌عامل لینوکس، امکانات خاصی همچون namespace ها، cgroup ها، chroot و سیستم‌های فایل Union را فراهم می‌کند که به کاربران و توسعه‌دهندگان این امکان را می‌دهد تا محیط‌های منزوی[2] شده‌ای برای اجرای برنامه‌ها و سرویس‌ها بسازند. در این پروژه، شما یک سیستم اجرا و مدیریت محفظه مانند Docker یا Podman پیاده‌سازی خواهید کرد. یکی از مزیت‌های Podman نسبت به Docker این است که معماری آن، بدون دیمون[3] است، به عبارتی یک دیمون (systemd service) اجرا نخواهد شد که وظیفه اجرا و مدیریت محفظه‌ها را به عهده بگیرد. این معماری بدین دلیل خوب است که نقطه تکی شکست[4] ایجاد نخواهد کرد. اگر دیمون داکر دچار مشکل شود، عملکرد تمام محفظه‌ها تحت‌تأثیر قرار خواهد گرفت. سیستم مدیریت محفظه‌ای که شما پیاده‌سازی می‌کنید نیز باید از همین معماری بدون دیمون پیروی کند.

یکی دیگر از تکنولوژی‌هایی که اخیراً در حوزه‌های مرتبط با سیستم‌عامل بسیار مورد توجه قرار گرفته است، eBPF است. برای مطالعه بیشتر می‌توانید به سایت این پروژه مراجعه کنید. در این قسمت لازم است شما با استفاده از این تکنولوژی، تمامی فراخوانی‌های سیستمی[5] که منجر به ایجاد یک namespace یا cgroup می‌شوند را نظارت کرده و گزارش (لاگ) آن‌ها را در یک فایل ذخیره کنید.

[1] Container
[2] Isolate
[3] Daemonless
[4] Single point of failure
[5] System call

## اهداف پروژه

- آشنایی با Namespace ها: درک و پیاده‌سازی انواع namespace ها در لینوکس، از جمله شبکه، فرایندها، کاربری، و mount.
- مدیریت منابع با cgroups: پیاده‌سازی محدودسازی منابع از جمله پردازنده مرکزی[6]، حافظه، I/O، و شبکه برای هر محفظه.
- منزوی‌سازی محیط‌ها با chroot: استفاده از ابزار chroot برای محدودکردن دسترسی فایل‌ها و دایرکتوری‌ها به داخل محفظه‌ها.
- ساخت سیستم‌های فایل Union: طراحی و پیاده‌سازی Union filesystem (مانند overlayfs) برای ایجاد محیط‌های فایل منزوی و ترکیب چندین دایرکتوری به‌صورت پویا.
- ایجاد یک سیستم مدیریت محفظه: طراحی و پیاده‌سازی یک سیستم مشابه Podman که قابلیت ایجاد، مدیریت، و نظارت بر محفظه‌ها را داشته باشد.
- تعامل با سیستم‌عامل لینوکس: استفاده از سیستم‌عامل به طور مستقیم و نوشتن کدهایی که با هسته لینوکس تعامل دارند.
- آشنایی با eBPF: استفاده از این تکنولوژی و درک قدرت آن در ایجاد پویای رفتار دلخواه در فضای هسته.

[6] CPU

## صورت پروژه

این سیستم یک فایل دودویی[7] را به‌عنوان ورودی دریافت کرده و آن را به‌صورت محفظه اجرا می‌کند. محفظه‌هایی که اجرا می‌شوند، به‌صورت پیش‌فرض باید فضای PID، UID، GID، mount، و Hostname مخصوص به خودشان را داشته باشند. به عبارتی، وقتی یک محفظه اجرا می‌شود، کاربری که درون محفظه قابل‌مشاهده است، لزوماً با کاربری که محفظه را اجرا کرده است یکسان نیست. همچنین PID پردازه‌های درون محفظه، با PID که روی میزبان[8] قابل‌مشاهده است یکی نیست. همچنین هنگامی که یک دستگاه ذخیره‌سازی در یک مسیر mount می‌شود، این mount نباید داخل محفظه قابل‌مشاهده باشد. همچنین hostname محفظه باید متفاوت از میزبان باشد و نیز تغییرات hostname درون و بیرون از محفظه روی‌هم تأثیر نگذارند. همچنین پردازه‌هایی که درون یک محفظه اجرا می‌شوند به طور پیش‌فرض نمی‌توانند با پردازه‌های درون محفظه‌های دیگر و نیز پردازه‌های روی میزبان ارتباط داشته باشند. (ارتباطات بین پردازه‌ای[9])

سیستم شما باید قابلیت‌های زیر را داشته باشد:

- سیستم مدیریت محفظه شما باید از این قابلیت پشتیبانی کند که تعدادی محفظه (برخلاف حالت پیش‌فرض) بتوانند با مکانیزم ارتباطات بین پردازه‌ای با یکدیگر ارتباط برقرار کنند.
- سیستم مدیریت محفظه شما، باید قابلیت اضافه‌کردن mount اشتراکی را داشته باشد، به این معنی که یک دستگاه (به طور مثال /dev/sdc) هر گاه mount شد، این mount به محفظه‌های مشخص شده منتشر[10] شود.
- سیستم مدیریت محفظه شما، باید قابلیت کنترل منابع استفاده شده توسط محفظه‌ها را داشته باشد، این منابع شامل زمان استفاده از پردازنده مرکزی[11]، حافظه، و I/O است.
- این سیستم برای حداقل کردن context switch، این قابلیت را دارد (به‌صورت پیش‌فرض فعال نیست، بلکه قابل فعال‌سازی است) که محفظه‌ها را، یا به عبارت دقیق‌تر پردازه اصلی درون یک محفظه را به‌صورت RR روی یک هسته پردازشی قفل کند، به‌طوری‌که آن محفظه فقط روی همان هسته اجرا شود، مثلاً اگر ۴ هسته پردازشی روی سیستم شما وجود داشته باشد، محفظه اولی که پس از فعال‌سازی این قابلیت اجرا می‌شود روی هسته اول، محفظه دوم روی هسته دوم و ... قفل خواهند شد.
- محفظه نباید فضای فایل سیستم میزبان را ببیند، بلکه باید یک فضای فایل سیستم جداگانه ببیند.
- پیاده‌سازی امکاناتی برای نظارت بر وضعیت محفظه‌ها و مدیریت منابع اختصاص‌داده‌شده به آن‌ها.
- قابلیت اعمال محدودیت‌ها بر روی منابع مختلف مانند حافظه، پردازنده، و I/O.
- تمام فراخوانی‌های سیستمی مربوط به ایجاد و حذف ns و cgroup ها نظارت شود.

[7] Binary
[8] Host
[9] Inter-process communication
[10] Propagate
[11] CPU time

## گام‌های پروژه

۱. **آشنایی با مفاهیم و ابزارهای سیستم‌عامل لینوکس**: مطالعه و آزمایش مفاهیم پایه مانند namespace ها، cgroup ها، و chroot.

۲. **پیاده‌سازی محیط منزوی برای محفظه‌ها**: شروع با ایجاد یک محیط منزوی برای فرایندهای مختلف با استفاده از namespace ها و chroot.

۳. **مدیریت منابع محفظه‌ها با cgroup ها**: توسعه سیستم‌هایی برای محدودکردن منابع در دسترس برای هر محفظه.

٤. **مدیریت محفظه‌ها**: ایجاد امکاناتی برای مدیریت، نظارت و کنترل محفظه‌ها، از جمله توقف، راه‌اندازی مجدد، و حذف.

٥. **ایجاد رابط کاربری**: طراحی یک رابط کاربری ساده برای مدیریت کانتینرها، شامل فرمان‌هایی مانند run، start، list، و status.

## بخش امتیاز مازاد

۱. باید این قابلیت را پیاده‌سازی کنید که یک محفظه، با همان وضعیت در حال اجرا متوقف شود، و سپس در زمان دلخواه، شروع به کار کرده و کار خود را از همان نقطه‌ای که متوقف شده بود، ادامه دهد. توجه داشته باشید که این قسمت باید با استفاده از cgroups freezer پیاده‌سازی شود.

۲. با استفاده از مفهوم Union filesystem چیزی شبیه به تصاویر داکر[۱۲] بسازید که سیستم قابلیت اجرای آن را داشته باشد، توجه کنید که افزودن این قابلیت نباید از اجرای فایل‌های دودویی جلوگیری کند.

[12] Docker images

## گزارش‌ها

دو گزارش بایستی برای این پروژه تحویل داده شوند که گزارش فاز اول شامل:

- **شرح مفاهیم**
- **معرفی ابزارها و محیط اجرایی**

و گزارش نهایی نیز باید شامل موارد زیر باشد:

- **موارد مطرح شده در گزارش فاز اول**
- **نحوه پیاده‌سازی**: توضیح دقیق از نحوه پیاده‌سازی هر بخش از سیستم، از جمله namespace ها، cgroup ها، و chroot.
- **کد منبع**: کد منبع کامل سیستم که به طور واضح و مستند، به همراه توضیحات باشد.
- **نتایج آزمایش‌ها**: گزارشی از آزمایش‌هایی که بر روی سیستم انجام شده است، شامل تست عملکرد و پایداری.
- **نقد و بررسی سیستم**: تحلیل مشکلات و محدودیت‌هایی که در طول پروژه با آن‌ها روبرو شده‌اید و پیشنهادهایی برای بهبود سیستم.

## بارم‌بندی

- ۴۰ نمره ← گزارش‌ها و مستندات
- ۶۰ نمره ← پیاده‌سازی و ارائه نهایی
- ۲۵ نمره ← امتیازهای مازاد
- مجموع نمرات از ۱۰۰ نمره محاسبه شده که به بارم نهایی مربوط به پروژه تبدیل خواهد شد.

## نکات تکمیلی

- موعدهای تعیین شده برای تحویل گزارش‌های دو فاز پروژه، مطابق تاریخ‌های تنظیم شده در تقویم درس هستند.
- تاریخ جلسه(های) ارائه پروژه در انتهای ترم، و پس از تحویل گزارش نهایی اطلاع‌رسانی خواهد شد.
- در صورت وجود هرگونه ابهام، می‌توانید با این آدرس ایمیل در ارتباط باشید.

**موفق باشید**