باسمه تعالی

آزمایشگاه سیستم عامل

پروژه شماره‌ی دو: فراخوان سیستمی

تاریخ تحویل: ۱۶ آبان

اهداف این پروژه

- آشنایی با پیاده‌سازی یک فراخوان سیستمی در هسته‌ی لینوکس
- آشنایی با لیست پیوندی[1] هسته‌ی لینوکس
- آشنایی با ساختار داده task_struct

شرح آزمایش

در این آزمایش تعدادی فراخوان سیستمی برای ایجاد یک جدول هش[2] در هسته‌ی لینوکس پیاده‌سازی خواهیم‌کرد. و مانند هر جدول هش دیگر، باید روشی برای جلوگیری از تصادم[3] پیش‌بینی کرده‌باشیم که آن استفاده از لیست پیوندی، و در این مورد خاص لیست پیوندی هسته‌ی لینوکس است.

نحوه‌ی اضافه‌کردن فراخوان سیستمی

برای انجام این کار مستندات زیادی در اینترنت و دیگر منابع موجود است. در این قسمت از پروژه شما باید به جستجوی روش انجام این کار بپردازید و با آزمودن و یافتن روش درست اضافه‌کردن فراخوان سیستمی، مراحل انجام کار را در گزارش خود مختصرا بیان نمایید.

فراخوان‌های سیستمی پروژه

در این قسمت قصد داریم فراخوان‌های سیستمی موردنیاز برای ساخت جدول هش مدنظر را پیاده‌سازی کنیم. در هسته‌ی لینوکس فایل‌ها[4] -به عنوان منابع سیستم- با file descriptor مربوطه‌شان شناخته می‌شوند و پردازه‌ها بدین وسیله به فایل دسترسی می‌یابند. می‌خواهیم یک جدول هش بسازیم که با گرفتن مسیر کامل[5] فایل،

---

[1] Linked List

[2] Hash Table

[3] Collision

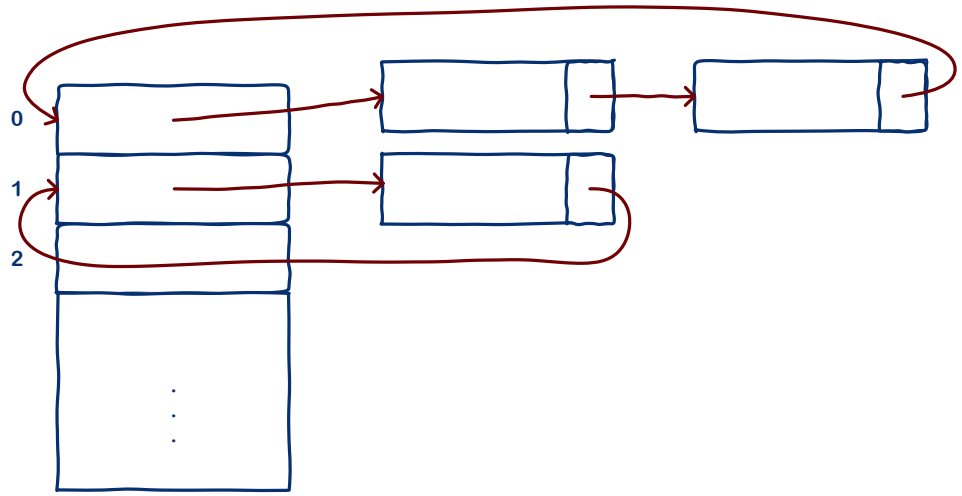
[4] "فایل" عنوانی است کلی‌تر از آنچه از در اصطلاح عادی فایل نامیده می‌شود.

[5] absolute path

شناسه‌ی پردازه‌هایی که به آن دسترسی دارند و مشخصه فایل[۶] را به ما بدهد. فایل‌هایی که **در حال حاضر** جزء منابع در دسترس هسته هستند باید در این جدول هش دیده‌شوند، یعنی فایل‌هایی که به‌ازای پردازه(ها)ای file desctiptor مشخص دارند.

پردازه‌های سیستم در ساختارداده task_struct[۷] تعریف می‌شوند. فایل‌های هر پردازه نیز جزء مواردی هستند که در این ساختار ذخیره می‌شود. با پیمایش تمامی پردازه‌های سیستم file descriptorهای موجود را استخراج کرده و مسیر کامل متصل به هر کدام را بیابید. نهایتا مسیر را به عنوان کلید هش و شناسه پردازه (pid) و file descriptor را به عنوان مقادیر در جدول هش خود ذخیره کنید.

بسته به اندازه جدول هش، تابع هش و ورودی‌ها میزانی از تصادم رخ می‌دهد، برای نگهداری مقادیری که کلیدهای آنها با هم تصادم دارند به کمک لیست‌پیوندی هسته، ساختارداده‌ای تعریف کنید و مقادیر را در آن ذخیره کنید. (مانند شکل زیر)



شبه کد تابع هش مدنظر در مقابل آمده است:

```
ans = 0
for each character
       ans = ans * 33 + ascii code of character
ans = ans mod table_size
```

در این تمرین، باید فراخوان های سیستمی زیر را پیاده‌سازی کنید:

- `asmlinkage long  sys_init_hash_table(unsigned int table_size)`
  این تابع با فراخوانی توابع کمکی دیگر، جدول هش با سایز آرگومان ورودی می‌سازد و آن را در هسته چاپ می‌کند

---

[6] file descriptor

[7] تعریف‌شده در /include/linux/sched.h

- `asmlinkage long  sys_show_pid_fd(char *path)`

این تابع با گرفتن آرایه‌ای از کاراکترها به عنوان مسیر، تمام pid و fdهای مرتبط با آن را در هسته چاپ می‌کند

- `asmlinkage long  sys_free_hash_table(void)`

این تابع حافظه‌ی اختصاص داده‌شده به جدول هش ما را آزاد می‌کند

علاوه بر این توابع که از نوع فراخوان سیستمی هستند، سایر توابع کمکی موردنیاز را براساس تشخیص خود پیاده‌سازی نمایید، هم‌چنین تابع main مناسب برای تست برنامه خود تهیه‌کنید تا از درستی نتیجه آن مطمئن شوید.

پرسش‌ها

در گزارش خود به سوالات زیر پاسخ دهید:

1. اندیس‌های 0 و 1 و 2 مشخص‌کننده‌ی کدام file descriptor هاست؟
2. file descriptor ها بجز فایل‌های واقعی موجود در سیستم به چه نوع ورودی یا خروجی‌های دیگری متصل هستند؟ به بیان دیگر انواع ورودی یا خروجی‌هایی که به کمک file descriptor ها در سیستم شناخته می‌شوند را بیان نمایید.
3. در مورد چیستی و کاربرد دو نوع مسیر زیر توضیح دهید

الف) /dev/null

ب) /dev/ttyX[8]

نکات پروژه

- منظور از لیست‌پیوندی هسته‌ی لینوکس struct list_head است.
- fd های مربوط به یک پردازه **اندیس‌های** fdtable آن هستند.
- حافظه‌ی هسته محدود است و نمی‌توانید تمامی مسیرهای موجود در کرنل را با اشغال‌کردن فضای هسته ذخیره سازید، برای رفع این مسئله به جای اشغال فضای جدید از ذخیره اشاره‌گر به ساختارهای موجود بهره بگیرید.
- در جدول هش ممکن است به ازای یک کلید چند مقدار داشته باشیم، این مقادیر را مانند سایر گره‌های لیست‌پیوندی در ادامه لیست ذخیره‌کنید، و یا بخش امتیازی: برای کلیدهایی که چندین مقدار دارند لیست‌پیوندی دوبعدی بسازید و مقادیر با کلیدهای عینا یکسان را در بعد دوم لیست‌پیوندی ذخیره‌نمایید.

[8] که X یک عدد است

توضیح اضافه: اینکه به ازای یک کلید می‌توانیم چند مقدار داشته‌باشیم به معنای دسترسی چند پردازه به یک فایل است، که این دسترسی با مکانیزم‌هایی مانند استفاده از قفل‌ها کنترل می‌شود، در فصل‌های بعدی درس با این مکانیزم‌ها عمیقا آشنا خواهید شد.

سایر نکات

- دقت داشته‌باشید که کد نوشته‌شده در کرنل لینوکس مانند سایر کدها نیاز به خوانایی، تمیزی، ماژولار بودن و... دارد و عدم رعایت این موارد منجر به کسر نمره می‌گردد.
- طبیعت پروژه‌های آزمایشگاه به گونه‌ای است که ممکن است با مشکلات پیش‌بینی‌نشده مواجه شوید. دستیاران آموزشی در رفع این مشکلات به شما کمک خواهندکرد، ولی مسئولیت انجام درست پروژه به عهده‌ی خود شما است. بنابراین توصیه میشود که پروژه را زود شروع کنید.
- حتما درجلسه‌ی توجیهی حضور داشته باشید. نکاتی که در کلاس یا فروم مطرح می‌شوند جزء پروژه هستند.
- پروژه‌های آزمایشگاه باید در گروه‌های سه نفره انجام شوند. تمام اعضای گروه باید روی تمام قسمت‌های پروژه تسلط داشته باشند و هریک از اعضا متناسب با میزان تسلط نمره دهی خواهد شد.

موفق باشید