|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |

|  |
| --- |
| **دانشگاه تهران**  **دانشکده مهندسی برق و کامپیوتر**  **آزمایشگاه سیستم عامل**  **آزمایش دوم** |

|  |  |
| --- | --- |
| نام و نام خانوادگی | مونا محدث مجتهدی-ریحانه احمدپور-آناهیتا مشتاق کهنمونی |
| شماره دانشجویی | 810198473-810198494-810198557 |
| تاریخ ارسال گزارش | 28/1/1401 |

فراخوانی سیستمی

مقدمه

**1)**فایل های تشکیل دهنده ULIB شامل printf.c,usys.s,ulib.c و umalloc.c می باشد.فایل های .c شامل کتابخانه های سطح کاربرمانند free,malloc,printf و... میباشد.

توابع printf.c:

Putc: این تابع یک کاراکتر را در مقصد که توسط یک file descriptor مشخص شده،چاپ میکند.در آن از

سیستم کال write استفاده شده است.

Printint: تابعی است که یک مقدار integer را در مقصد که توسط یک file descriptor مشخص شده،چاپ

میکند که برای این منظور تمام ارقام بافر را توسط تابع putc درمقصد چاپ میکند.

Printf: این تابع یک string شامل اعداد و کاراکترها را در مقصد چاپ میکند که برای این منظوراز توابع

Putc و printint استفاده میکند.

ulib.c:

Stat: این تابع اطلاعات مربوط به یک فایل را در قالب ساختار statبرمیگرداند.دراین تابع از سیستم کال های

Close,open و fstat استفاده میشود.

شامل توابع atoi,strlen,strcmp,strcpy و ... میباشد که توابع مربوط به کار کردن با استرینگ ها میباشد.

umalloc.c:

Free: این یک پوینتر را به عنوان ورودی میگیرد که این پوینتر به یک آدرس ازحافظه اشاره میکند و سپس این

بخش از حافظه را آزاد میکند.

Morecore:این تابع توسط سیستم کال sbrk ،حافظه پردازه را گسترش میدهد.

Malloc: برای تخصیص حافظه پویا با اندازه مشخص استفاده میشود که مقدار خروجی آن یک اشاره گر void

به حافظه تخصیص داده شده است.

Usys.S: این فایل شامل پوشاننده های سیستم کال ها به زبان اسمبلی میباشد.در این فایل یک تابع به نام SYSCALL(name) وجود دارد که ابتدا مقدار SYS\_name را در رجیستر eax میریزد و سپس یک وقفه با کد T\_SYSCALL تولید میکند.

**2)** روش اول:از طریق File system

لینوکس دارای API های دیگری است که از طریق pseudo-file system ها مانند /proc,/sys,/dev به هسته

دسترسی پیدا میکند.مثلا فایل سیستم /proc ارتباط بین فضای هسته و کاربر را فراهم میکند. /sys یک رابط برای هسته است که به طور مشخص ،یک نمای سیستم فایلی از اطلاعات و تنظیمات پیکربندی را ارائه میکند. /dev محل دیوایس فایل ها میباشد.

روش دوم:از طریق signal

سیگنال یک پیام بسیار کوتاه است که ممکن است به یک فرآیند یا گروهی از فرآیند ها ارسال شود.تنها اطلاعاتی که به فرآیند داده میشود معمولا عددی است که سیگنال را شناسایی میکند. سیگنال های استاندارد آرگومان ها یا سایر اطلاعات را منتقل نمیکنند.مجمموعه ای از ماکرو ها که نام آنها با پیشوند SIG شروع میشود برای شناسایی سیگنال ها استفاده میشود.سیگنال ها دو هدف اصلی را دنبال میکنند.1-برای آگاه ساختن یک فرآیند از وقوع یک رویداد خاص

2-مجبور کردن یک فرآیند به اجرای handler function آن سیگنال

روش سوم:استفاده از socket

در این حالت برنامه هایی که در سطح کاربر هستند میتوانند با استفاده از سوکت روی یک پورت خاص گوش دهند و اطلاعات را رد و بدل کنند.

ساز و کار اجرای فراخوانی سیستمی در xv6

**3)** خیر سطوح سیستم عامل از شماره 0 تا 3 نامگذاری شدند که در سیستم عامل xv6 تنها دو سطح 0 یعنی سطح کرنل و سطح کاربر وجو دارد. در فراخوانی سیستمی ،تله ها در سطح 3 یعنی DPL\_USER فعال میشوند چون فراخوانی های سیستمی توسط قطعه کدی در سطح کابر فراخوانی میگردد و اگر برنامه سطح کاربر سطح دسترسی مناسبی نداشته باشد ،دستور int باعث به وجود آمدن دستور int 13 میشود که یک خطای حفاظتی protectionاست اما بقیه تله ها که از سطح کرنل صدا زده میشوند که عموما مربوط به درایور های سیستم و حافظه سیستم می باشند ،باید از سطح دسترسی 0 فعال شوند.

**4)** به طور کلی دو پشته داریم که یکی برای سطح کاربر و دیگری برای سطح kernel است.هنگامی که میخواهیم سطح دسترسی که انجام داده این چه بوده و ادامه روند اجرای دستورات را از سر بگیریم.حال با توجه به اینکه داشتن این اطلاعات ضروری است ،نیاز داریم که هر بار با تغییر سطح دسترسی این اطلاعات را روی پشته پوش کنیم.به همین ترتیب زمانی که تغییر سطح نداشته باشیم ، نیاز نداریم که پوش کنیم.

**5)** توابع دسترسی argptr()و argint() و argstr() و argfd() هستند که به ترتیب ازargint() و fetchint()

و fetchstr() در فایل syscall.c و argfd() در فایلsysfile.c استفاده میکند.

Argint(): این تابع دو ورودی دارد شماره پارامتر به عنوان ورودی اول و ورودی دوم به صورت pass by reference که یک متغیر با int type است داده میشود که در واقع پارامتر در این متغیر(ورودی دوم) ذخیره میشود.این تابع با صدا زدن تابع fetchinit() و دادن مقدار (myproc()->tf->esp)+4+4\*n به عنوان ورودی اول آن ،آدرس پارامترخواسته شده را ساخته و محتوای آن را برمیگرداند.اگر باموفقیت انجام شود مقدار صفر و اگر پارامتر وجود نداشت خروجی 1- برمیگرداند.

Argptr: این تابع سه ورودی دارد. شماره پارامتر به عنوان ورودی اول و ورودی دوم به صورت pass by reference و سایز پارامتر به عنوان ورودی سوم داده میشود که محتوای پارامتر های به شکل پوینترمانند آرایه را در ورودی دوم ذخیره میکند.این تابع با صدا زدن argint() آدرس خانه اول این پارامتر از جنس اشاره گر را برمیگرداند. توجه شود که این تابع در صورتی که با موفقیت عملیات را انجام دهد مقدار صفر و در صورت شکست 1- برمیگرداند.

Argstr(): این تابع دو ورودی دارد شماره پارامتر به عنوان ورودی اول و ورودی دوم به صورت pass by reference داده میشود.این تابع برای مشخص کردن پارامتر های از جنس رشته است.ابتدا با استفاده از argint() آدرس خانه اول رشته را بدست میآورد و سپس با صدا زدن تابع fetchstr() رشته را در ورودی دوم ذخیره میکند.همانند قبلی ها در صورت موفقیت صفر و درغیر این صورت 1- برمیگرداند.

Argfd(): این تابع در فایل sysifle.c تعریف شده و سه ورودی دارد.ورودی اول شماره پارامتر ورودی دوم آن اشاره گری به توصیف کننده فایل یا همان file descriptor و ورودی سوم اشاره گری به آدرس ساختمان داده فایل است که پس از انجام عملیات تابع مقداردهی میشوند.

در تابع argptr پس از مشخص شدن آدرس شروع توسط argint اگر این تابع 1- برگرداند یعنی آدرس شروع نامعتبر بوده و عملیات با شکست مواجهه شده.سپس سه شرط چک میشود:1-سایز منفی نباشد2-آدرس شروع از سایز پردازه فعلی کمتر نباشد3-جمع آدرس شروع و سایز داده شده در ورودی سوم بیشتر از سایز این پردازه باشد.در صورت برقراری این سه شرط یعنی پارامتر معتبر نبوده و دسترسی ما به جای اشتباهی از استک صورت گرفته و اشاره گر به جایی غیر از قسمت حافظه اختصاص داده شده به این پردازه اشاره دارد پس عملیات شکست خورده و 1- برمیگرداند. اگر این باز ها چک نشوند ممکن است ما محتوای قسمت هایی از حافظه را بخوانیم که اشتباه هستند و بنابراین برنامه اشتباه انجام میشود و باممکن است به قسمتی از حافظه دسترسی پیدا کنیم و تغییری در آن ایجاد کنیم که در این صورت ممکن است باعث بروز خطا در پردازه های دیگر و یا افتادن در تله شویم.

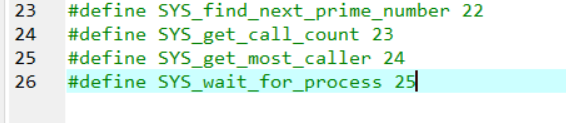
در فراخوانی سیستمی sys\_read() اگر این بازه ها چک نشوند برای مثال ممکن است محتوای فایلی که میخوانیم در قسمت نادرستی ذخیره شود که مخصوص این پردازه نیست و بنابراین هم در انجام عملیات پردازه فعلی هم دیگر پردازه ها مشکل ایجاد میشود و بخشی از اطلاعات حافظه از دست برود و یا در تله بیفتیم.

ارسال آرگومان های فراخوانی های سیستمی

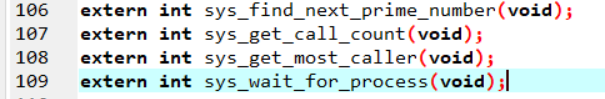
نحوه اضافه کردن فراخوانی های سیستمی

برای اضافه کردن ،چهار تا سیستم کال های جدید باید مراحل زیر انجام شود:

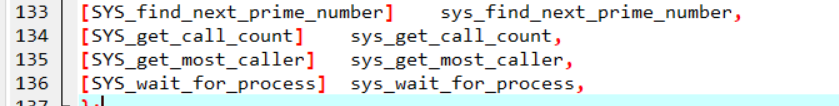
1-در فایل syscall.h یک عدد برای سیستم کال های جدیدمان تعریف میکنیم



-2در فایل prototype ،syscall.c توابع این سیستم کال ها را اضافه میکنیم.

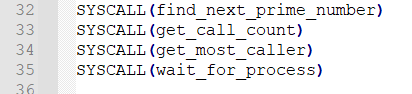


3- در فایل syscall.c یک پوینتر به سیستم کال ها اضافه میکنیم. در این فایل یک آرایه از pointer function ها داریم که به واسطه اعداد نسبت داده شده به سیستم کال ها،یک پوینتر به سیستم کال ها تعریف میکند.با اینکار هر زمان که یک سیستم کال را با شماره متناظر آن صدا کنیم،تابعی که سیستم کال به آن اشاره میکند هم صدا زده ایم.

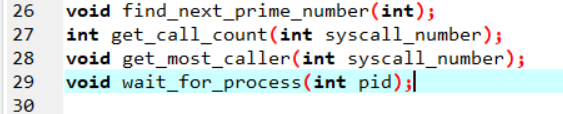


4-حال باید تابع سیستم کال ها را تعریف کنیم.بسته به اینکه چه سیستم کالی داریم،میتوان در دو فایل sysproc.c و sysfile.c انجام داد. (انجام ندادم)

-5در فایل usys.S باید بگوییم که عدد سیستم کال هایی که اضافه کردیم را در رجیستر eax بریزد.با اینکار،توسط دستور int $T\_SYSCALL میتوان به سیستم عامل رخ دادن سیستم کال را اعلام کرد.



6-در فایل user.h باید prototype توابع را بنویسیم که برنامه سطح کاربر به واسطه صدا کردن آن ها،به سیستم کال مورد نظر متصل شود.



7- در انتها باید یک فایل اضافه کنیم که در آن فایل کار مورد نظر را با صدا کردن تابعی که در user.h است انجام دهیم.همچنین باید در makefile،قسمت UPROGS که دستورات وجود دارند،دستورات مورد نظر اضافه کنیم و در قسمت EXTRA باید فایل های مورد نظر را قرار دهیم. (انجام ندادم)