

به نام خدا

پروژه xv6

فاز یک



محمد جواد زندیه

شماره دانشجویی : 9831032

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر

پروژه درس سیستم عامل 28/7/1400

getProcCount System Call

هدف : بدست آوردن تعداد process هایی را که در لحظه فراخوانی این فراخوانی های سیستمی وجود دارند (یعنی تمامی process هایی که در Unused state نباشند)

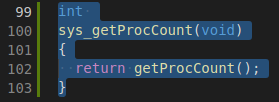
فایل هایی از کد سیستم عامل xv6 تغییر کرده اند تا این system call را به سیستم عامل اضافه کنیم :

1. syscall.h



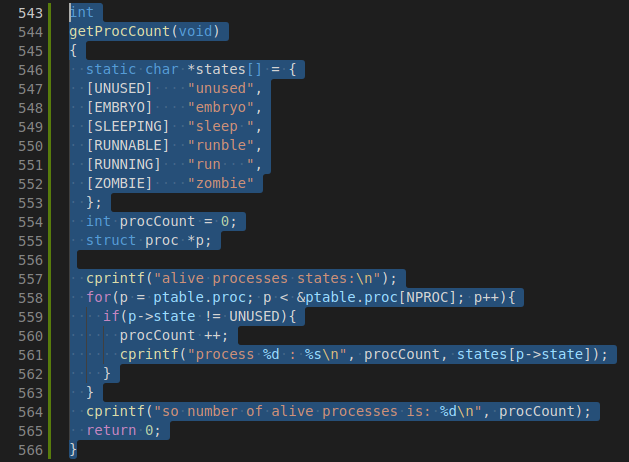
در این فایل شماره متناظر با system call ای را که میخواهیم تعریف کنیم set می کنیم.

1. sysproc.c



در این فایل system call خود را تعریف میکنیم و پیاده سازی را به تابع getProcCount می سپاریم.

1. proc.c



در این فایل همانطور که در مرحله قبل گفته شد به پیاده سازی انتظاری که از system call داریم می پردازیم. یک تعدادی اطلاعات اضافه هم به منظور تفهیم بهتر قرار داده شده است، از جمله اینکه هر یک از process ها در چه state ای قرار دارند. روش پیاده سازی از این قرار است که با پیمایش روی تمام process هایی که تاکنون در سیستم قرار گرفته اند آن process هایی را که در unused state هستند را لیست کرده و در نهایت تعداد آنها را بیان می کنیم.

1. syscall.c



با extern کردن تابع sys\_getProcCount در این فایل، این system call را به function pointer array سیستم اضافه می کنیم.

1. defs.h



دسترسی به function های فایل proc.c از این فایل صورت می گیرد و باید تابع پیاده ساز این فراخوانی را در این فایل تعریف کنیم.

1. user.h



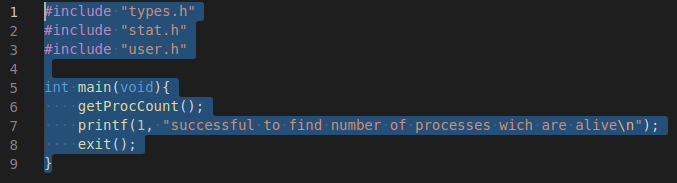
این فایل یک interface برای کاربر است و لیست توابع سیستمی در آن قرار می گیرد.

1. usys.S



یک فایل assembly است که می گوید عدد integer ای که برای هر system call در نظر گرفته شده است به ثبات eax منتقل شود.

1. getProcCountTest



یک فایل تست است برای ارزیابی درستی اضافه شدن system call به این سیستم عامل

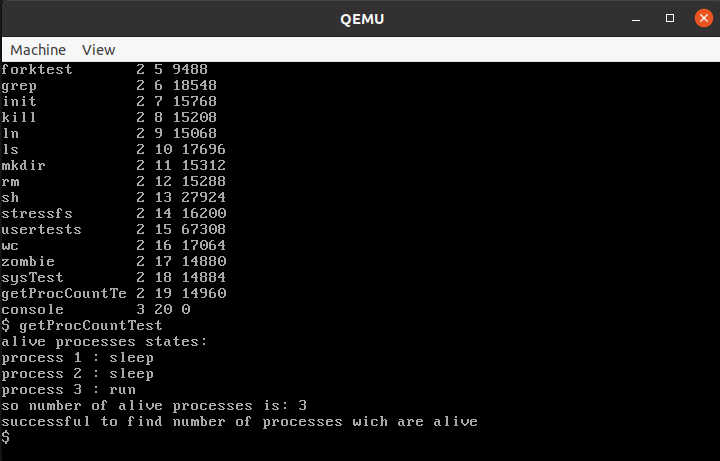
1. Makefile





قسمت های EXTRA, PROGS از این فایل را به صورت عکس داده شده تغییر می دهیم تا در لیست ای که توسط دستور ls نشان داده می شود از system call ها، این فراخوانی هم اضافه شود.

و اما اگر بخواهیم یک نمونه از صدا کردن این فراخوانی را تست کنیم:



همانطور که مشاهده می شود getProcCount در لیست system call ها افزوده شده است و صدا کردن آن هم میتوان دید که 3 تا process در سیستم وجود داشته اند که دوتای آنها در sleep state بوده اند و یکی هم که خود این process بود در run state قرار دارد.

getReadCount System Call

هدف : بدست آوردن تعداد دفعاتی که فراخوانی سیستمی Read توسط هر پردازه دیگر کاربر فراخوانی شده است از زمانی که کرنل بوت شده است.

فایل هایی از کد سیستم عامل xv6 تغییر کرده اند تا این system call را به سیستم عامل اضافه کنیم :

توضیحات بسیاری از تغییرات همانند قسمت اول است پس فقط به توضیح قسمت هایی که متفاوت عمل شده می پردازیم.

1. proc.h



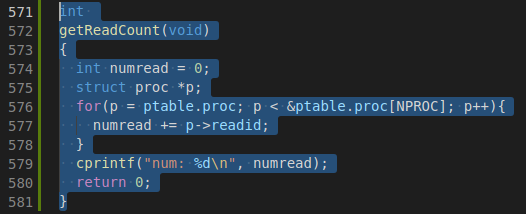
در این فایل یک struct به نام proc وجود دارد که مشخص می کند هر یک از process ها حاوی چه اطلاعاتی باشند. در این بخش می توان گفت که هر یک از process ها یک عدد که حاوی تعداد فراخوانی Read ای میباشد که تاکنون انجام داده را نگهداری کند.

1. proc.c



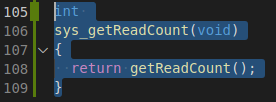
در این فایل مقدار تعداد فراخوانی را برابر صفر قرار میدهیم(مقدار دهی اولیه میکنیم).

این مقدار دهی در قسمت allocproc از برنامه صورت می گیرد یعنی جایی که هر process شروع به گرفتن حافظه از ram می کند.



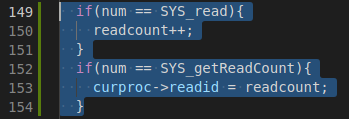
در این فایل همچنین به پیاده سازی تابع getReadCount می پردازیم و با iterate کردن روی تمام پردازه ها می توان تعداد فراخوانی های Read ای که تاکنون انجام داده اند را بدست آورد و جمع زد.

1. sysproc.c



در این فایل system call خود را تعریف میکنیم و پیاده سازی را به تابع getReadCount می سپاریم.

1. syscall.c



در این فایل، تابع syscall هر زمانی که یک system call صورت می گیرد اجرا می شود، پس می توان هر وقت که Read فراخوانی شد، به مقدار readid از پردازه کنونی یک واحد اضافه کرد.





با extern کردن تابع sys\_getReadCount در این فایل، این system call را به function pointer array سیستم اضافه می کنیم.

1. syscall.h



در این فایل شماره متناظر با system call ای را که میخواهیم تعریف کنیم set می کنیم.

1. defs.h



دسترسی به function های فایل proc.c از این فایل صورت می گیرد و باید تابع پیاده ساز این فراخوانی را در این فایل تعریف کنیم.

1. user.h



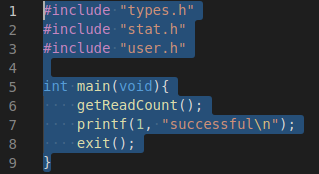
این فایل یک interface برای کاربر است و لیست توابع سیستمی در آن قرار می گیرد.

1. usys.S



یک فایل assembly است که می گوید عدد integer ای که برای هر system call در نظر گرفته شده است به ثبات eax منتقل شود.

1. getReadCountTest



یک فایل تست است برای ارزیابی درستی اضافه شدن system call به این سیستم عامل

1. Make file





قسمت های EXTRA, PROGS از این فایل را به صورت عکس داده شده تغییر می دهیم تا در لیست ای که توسط دستور ls نشان داده می شود از system call ها، این فراخوانی هم اضافه شود.

و اما اگر بخواهیم یک نمونه از صدا کردن این فراخوانی را تست کنیم:

با فراخوانی ls میتوان دید که هر دو system call مورد نظر ما قرار دارند.

با فراخوانی getReadCountTest قابل مشاهده است که نتیجه بیان می کند که 70 فراخوانی سیستمی Read توسط پردازه های ما از زمان بوت شدن انجام شده است.



