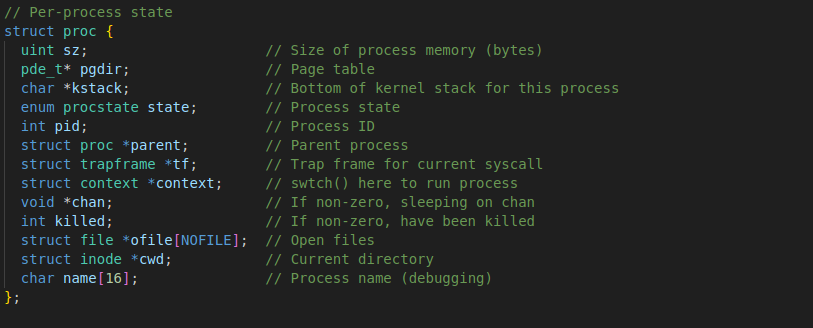
گزارش کار آزمایشگاه 3 آزمایشگاه سیستم عامل

1و2-

ساختار pcb در xv6 :



در شکل کتاب pcb شامل بخش های زیر است:

Process state

Process number

Program counter

Register

Memory limit

List of open files

حال همانطور که می بینید ofile معادل list of open files است.state معادل process state است.pid معادل process number است.program counter توسط ساختار trapframe مدیریت می شود. Memory limit توسط متغیر sz مشخص می شود.

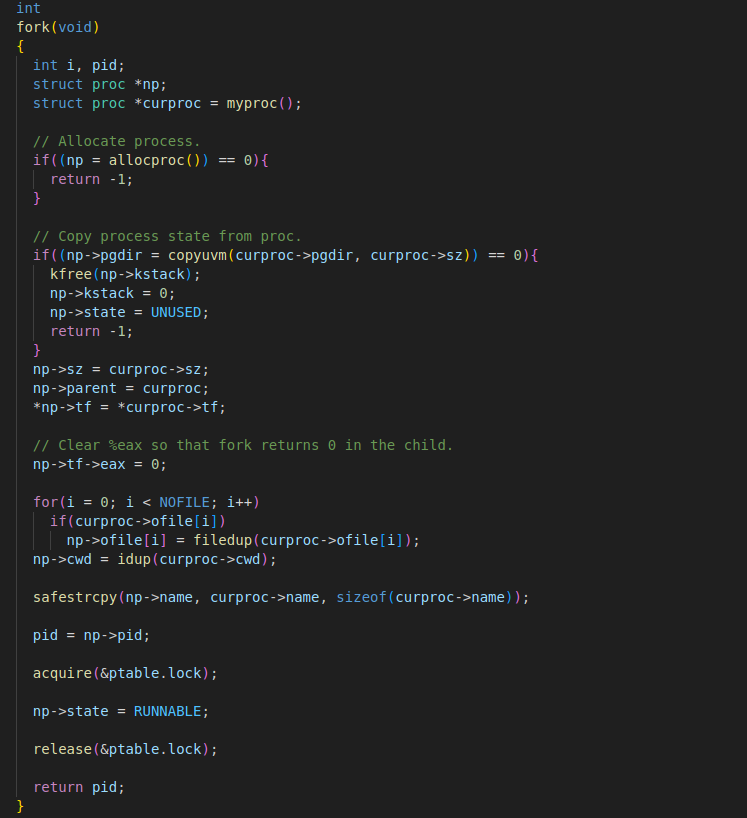
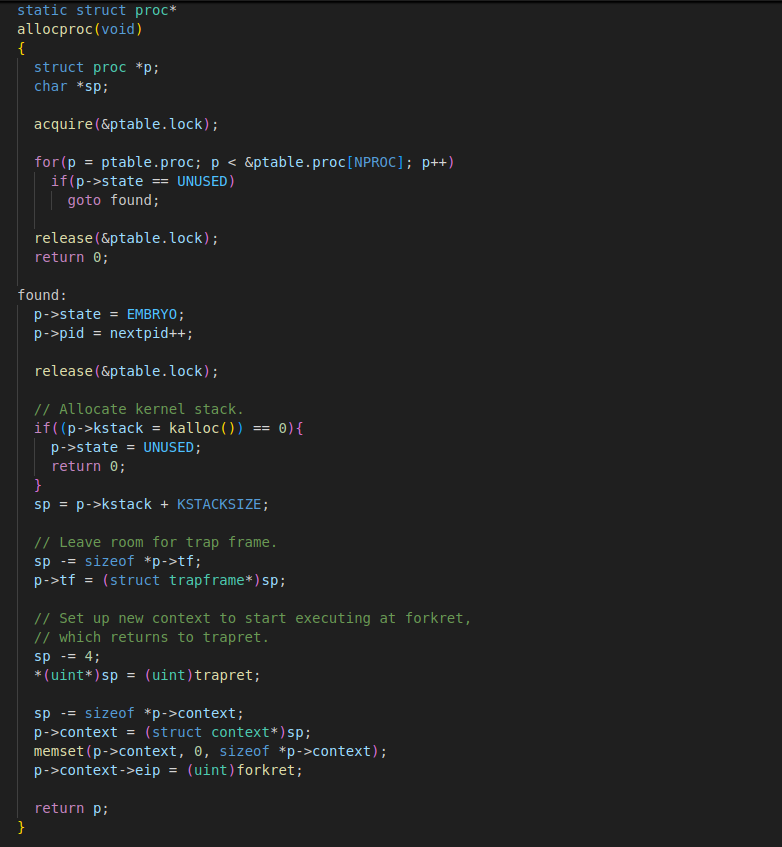
در فایل proc.h ، enum procstate را مشاهده می کنیم:



New معادل embryo می باشد.runnable یعنی پراسس هنوز ران نشده و قابل ران شدن است پس معادل ready است.sleepingمعادل waiting است.استیت running معادل هم نام خودش در مرجع است.استیت zombie یعنی پراسس ترمینیت شده ولی هنوز parent متوجه نشده است.unused یعنی اسلات پراسس خالی است.zombie را تا حدودی می توان معادل terminate گرفت.

3-

به ساختار دو تابع fork و allprocتوجه کنید:



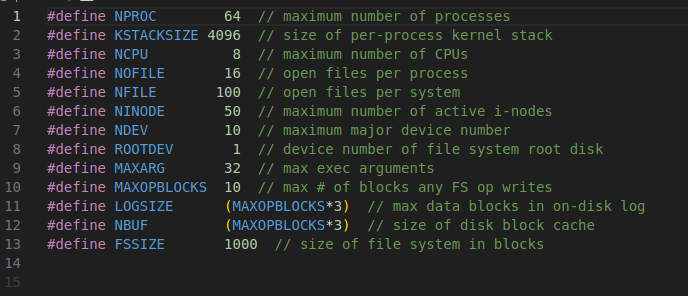
Allproc در fork صدا می شود.

در تابع allproc ابتدا ptable قفل می شود تا در هنگام تغییر ptable ، ptable تغییر نکند.سپس در ptable دنبال اسلات بلا استفاده می گردد.اگر پیدا شد اسلات را به پراسس جدید می دهد و وضیعت پراسس به embryo می رود.pid نیز آپدیت می شود.یک فضای کرنل استک به پراسس اختصاص می یابد.برای cpu context و trap frame نیز مقادیر دیفالت تعیین می شود.

حال در تابع fork پس از ساخت اولیه پراسس وضیعت به runnable معادل ready تغییر می کند.

4-

در فایل param.h داریم:



همانطور که از کامنت مشخص است سقف تعداد پردازه با کانستنت NPROC مشخص شده که 64 است.

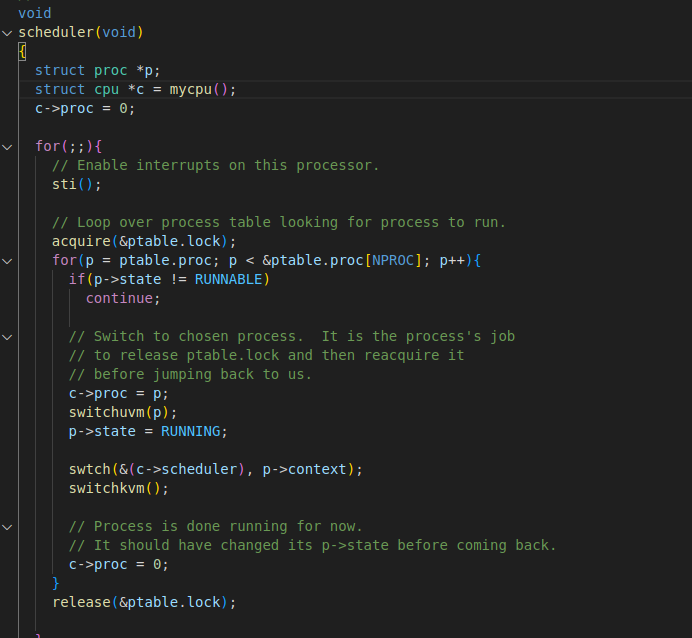
اگر به کددallproc دقت کنید در صورتی که اسلات بلااستفاده یافت نشود تابع allproc عدد 0 را برمی گرداند که در تابعfork بعد از دریافت این عدد خودش-1 برمی گرداند که یعنی پراسس جدید ساخته نشده است.در موقع کد سطح کاربر ما این خطا را معمولا مدیریت می کنیم و اگر pid منفی یک بود یعنی پراسس جدید ساخته نشده است.

5-

در سیستم های چند پردازنده ای به دلیل مسائلی مانند race condition که ممکن است دیتاهای مهم و حساسی مانند وضیعت هر پردازه و اولویت پردازه و غیره توسط دو پراسس تغییر کند تناقضی پیش بیاید.همچنین در زمان تصمیم گیری برای انجام پردازه ی بعدی نباید در میانه ی کار جدول پردازه ها تغییر کند.

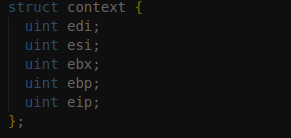
در سیستم های تک پردازنده ای به دلیل اجرای سریالی دستور ها لزومی به این کار در اغلب موارد نیست مگر اینکه interrupt بیاید و این interrupt ممکن است جدول پردازه ها را تغییر دهد بنابراین یا باید interrupt ها نافعال شوند یا جدول پردازه ها قفل شود.

6-



همانطور که در کد نیز مشخص است اگر پردازه runnable نباشد به iteration بعدی برای اجرا می رود.

7-



8-

رجیستر eip مخفف extended instruction pointer است .این رجیستر آدرس دستورالعمل بعدی را در کد برنامه نگهداری می‌کند. این رجیستر مشخص می‌کند که CPU در چه نقطه‌ای از برنامه باید به اجرای دستورالعمل‌ها ادامه دهد.

مقدار رجیستر های مهم ابتدا ذخیره می شوند و سپس به استک منتقل می شوند و در آخر در ساختار context کپی می شوند.

9-

بدیهتا زمانبندی پردازه ها مدیریت نمی شد.یک پردازه به طور نامحدود روی cpu اجرا می شد و باعث starvation بقیه پردازه ها می شد.

از آن طرف به طور کلی فعال نشدن وقفه باعث می شود که مدیریت منابع مشترک مدیریت ورودی خروجی ها و .. به درستی انجام نگیرد.

10-

پس از انجام عملیات گفته شده در هر 10 ثانیه حدود 1000 عدد چاپ شد.بنابراین هر 10 میلی ثانیه وقفه ی تایمر صادر می شود.

11-

تابع yield باعث انجام شدن گذار interrupt می شود.این تابع باعث تغییر حالت پراسس فعلی به runnable و آزاد کردن پردازنده باعث می شود پردازنده امکان رسیدن به وقفه ها یا انتخاب پراسس جدید برای scheduler می شود.

12-

با توجه به آن که هر وقفه ی تایمر نشان دهنده یک کوانتوم زمانی می باشد بنابراین هر کوانتوم زمانی 10 میلی ثانیه است.

13-

در سیستم‌عامل xv6، تابع wait برای منتظر ماندن تا اتمام کار یک پردازه‌ی فرزند استفاده می‌شود. این تابع در نهایت از مکانیزمی برای انتظار استفاده می‌کند تا زمانی که وضعیت پردازه‌ی فرزند zombie تغییر کند.

با قفل کردن ، تمام پردازه‌های فرزند پردازه جاری بررسی می‌شوند. اگر پردازه‌ای با وضعیت zombie یافت شود، منابع مربوط به آن آزاد شده و پردازه از جدول پردازه‌ها حذف می‌شود.اگر هیچ پردازه‌ی فرزندی با وضعیت zombieیافت نشود، تابع sleepفراخوانی می‌شود. تابع sleepپردازه جاری را به حالت sleeping منتقل می‌کند و منتظر می‌ماند تا وضعیت پردازه فرزند تغییر کند.

14-

انتظار برای رویدادها،همگام‌سازی،مدیریت منابع،ورودی/خروجی،زمان‌بندی،و مدیریت پردازه‌ها

15-

تابع wakeup نقش کلیدی در بیدار کردن پراسس هایی دارد که به حالت sleeping رفته اند.

16-

باعث تغییر وضیعت از وضیعت sleeping به runnable می شود.

17-

تابع wakeup1 می تواند علاوه بر wakeup باعث انجام این گذار شود.