|  |  |  |
| --- | --- | --- |
|  | **آزمایشگاه سیستم عامل** | آزمایش یک |
| استاد: **دکتر مهدی کارگهی** | اعضای گروه: **فرزین اسدی کاوان** |
|  |  |

Repo URL:

https://gitlab.com/btk-os/os-lab1.git

Last Commit ID:

57794dafdd227445358d786ce283cd5d65afec97

# **آشنایی با سیستم عامل xv6**

## **۱. ‫معماری سیستم عامل xv6 چیست؟ چه دلایلی در دفاع از نظر خود دارید؟**

معماری سیستم عامل xv6 بر پایه monolithic kernel است (همانگونه که اکثر سیستم عاملهای مبتنی بر UNIX پیاده سازی شده اند)، زیرا تمامی سیستم عامل در هسته قرار گرفته و تمامی کارها در سیستم عامل با تمامی اختیارات سخت افزاری انجام میشود. این موضوع باعث میشود که پیاده سازی سیستم عامل ساده تر شود اما ممکن است عواقبی داشته باشد. به طور مثال، در صورتی که یک اشتباه در سیستم عامل رخ دهد، این اشتباه میتواند بسیار مخرب باشد و منجر به توقف فعالیت سیستم عامل شود.

**۲. یک پردازه در سیستم عامل xv6 از چه بخش هایی تشکیل شده است؟ این سیستم عامل به طور کلی چگونه پردازنده را به پردازه های مختلف اختصاص می دهد؟**

به طور کلی یک پردازه در xv6 شامل حافظه که این حافظه خود شامل دستورات، داده و پشته است و یک حالت برای هر پردازه که به صورت خصوصی در اختیار هسته است، میباشد. همچنین هر پردازه با یک شناسه برای هسته سیستم عامل قابل شناسایی است. به طور کلی، سیستم عامل از روش اشتراک گذاری زمان و زمان بندی (time-sharing) بین پردازه ها استفاده میکند، یعنی بطور مداوم و بدون اینکه کاربر متوجه شود (transparency)، پردازنده های موجود را بین پردازه های آماده اجرا شدن پخش میکند و زمانی که یک پردازه در حال اجرا نیست، سیستم عامل مقدار ثباتهای مربوط به پردازنده را ذخیره کرده تا برای اجرا مجدد آماده باشد.

**3.مفهوم file descriptor در سیستم عامل های مبتنی بر UNIX چیست؟عملکرد pipe در سیستم عامل**

**xv6 چگونه است و بطور معمول برای چه هدفی استفاده میشود؟**

**یک عدد صحیح هستند که id یک آبجکت سیستم هست که برنامه میتواند در ان بنویسد یا در آن بنویسید.**

**برای برنامه فرقی ندارد که دارد با یک فایل یا یک پایپ یا یک io ارتباط برقرار میکند و مدیریت کردن**

**گردن سیستم عامل هست. پایپ ها در این سیستم یک جفت file descriptor هست که از طریق یکی**

**نوشتن در پایپ صورت میگیرد و از طریق یکی خواندن از پایپ برای ارتباط کنترل شده بین برنامه ها**

**استفاده میشود.**

**4. فراخوانی های سیستمی exec و fork چه عملی انجام میدهند؟ از نظر طراحی، ادغام نکردن این دو**

**چه مزیتی دارد؟**

**دستور fork یک فرآیند مانند فرآیند پدر می سازد و به همان اندازه حافظه اختیارمیکنند.**

**دستور exec فرآیند کنونی را جایگزین می کند(به جز file descriptor) وقتی این دو دستور جدا باشند**

**می توان یک فرآیند را fork کرد و بعد از آن ورودی و خروجی هایش را تغییر داد و بعد execرا با آن**

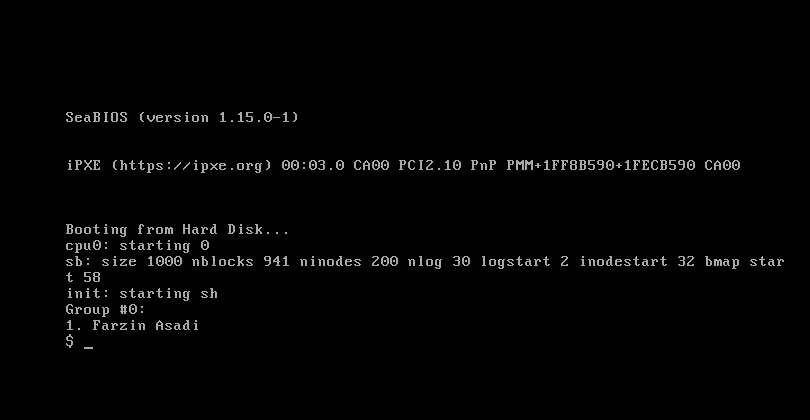
**ورودی و خروجی خاص صدا زد. اگر دو دستور یکی بودند مجبور بودیم ساز و کاری سخت تر برای**

**اینکار اتخاذ کنیم.**

**اجرا و اشکال زدایی**

**اضافه کردن یک متن به Message Boot**

در این بخش نام اعضای گروه را هنگام اجرای اولیه سیستم عامل روی صفحه نمایش چاپ می کنیم:

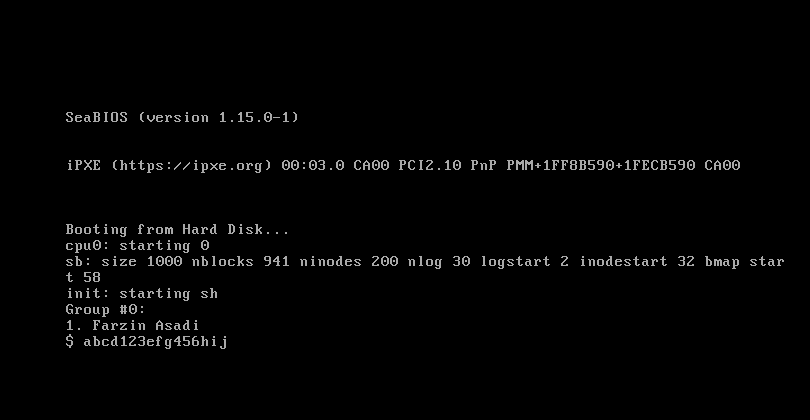


**اضافه کردن چند قابلیت به کنسول xv6**

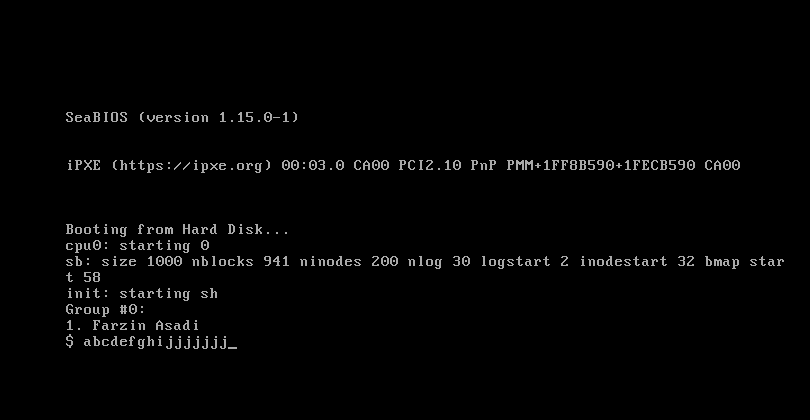
در این بخش از پروژه، ۳ قابلیت جدید به کنسول اضافه می کنیم:

**۱. Ctrl+N:**

این قابلیت اعداد را حذف میکند :

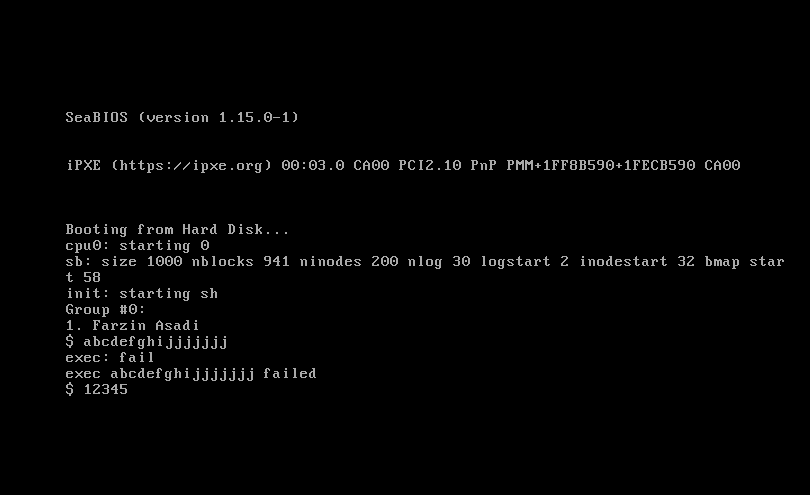


Ctrl + N:

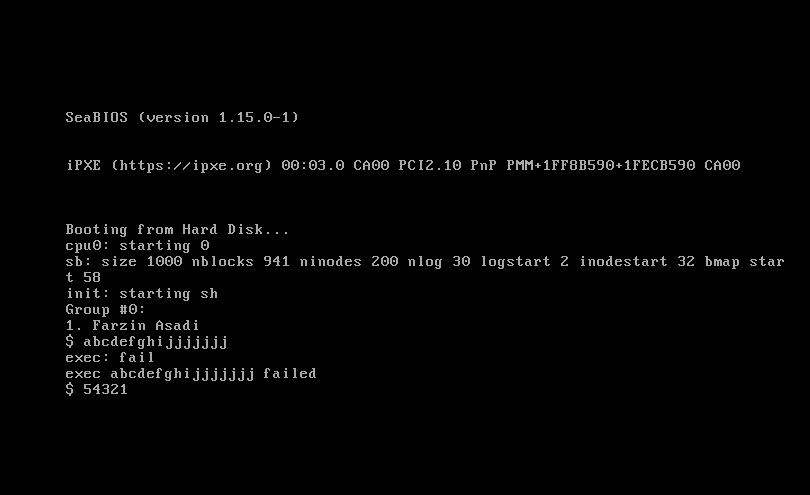


**۲. Ctrl+R:**

این قابلیت متن را reverse میکند:

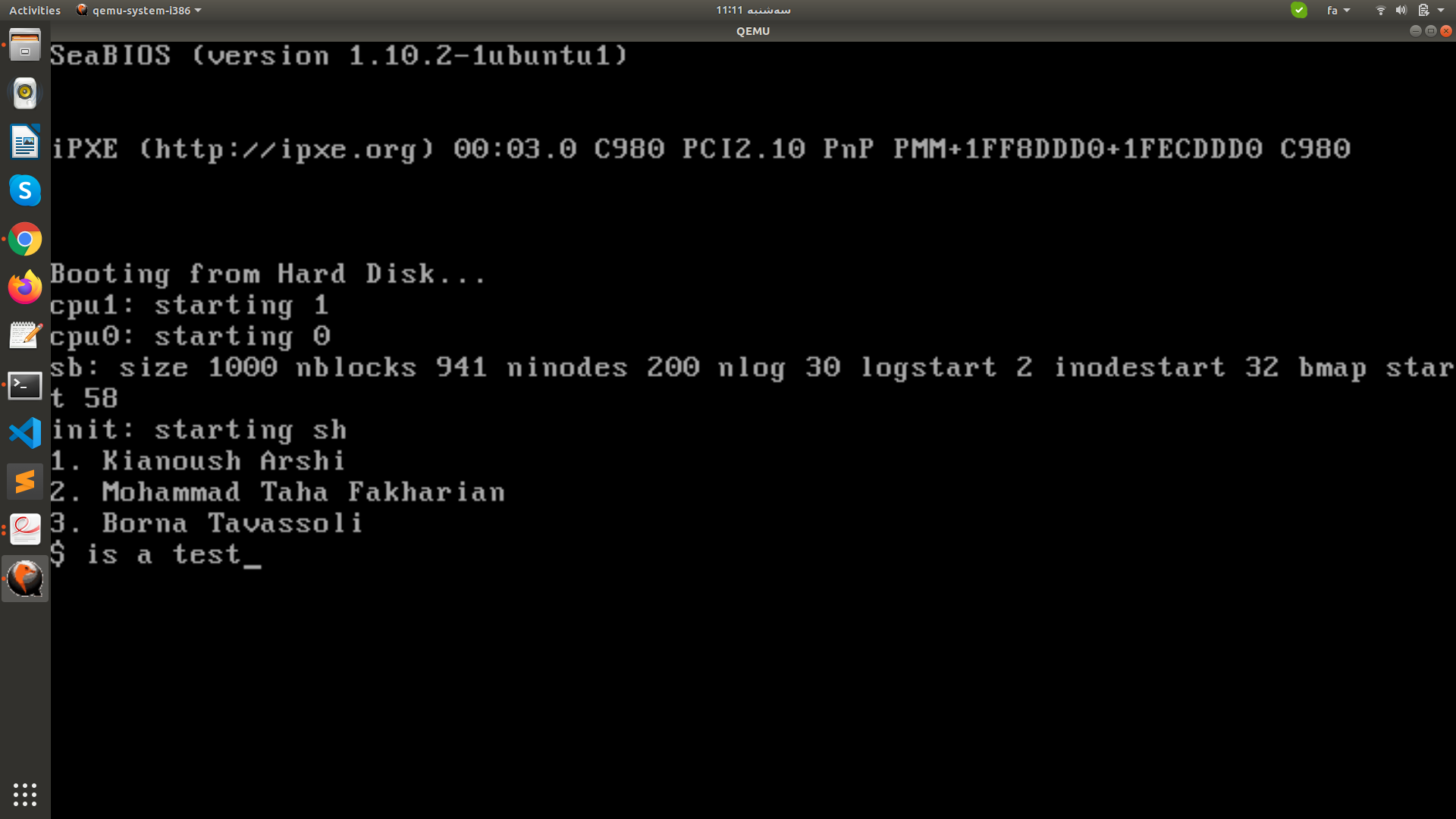


Ctrl + R:

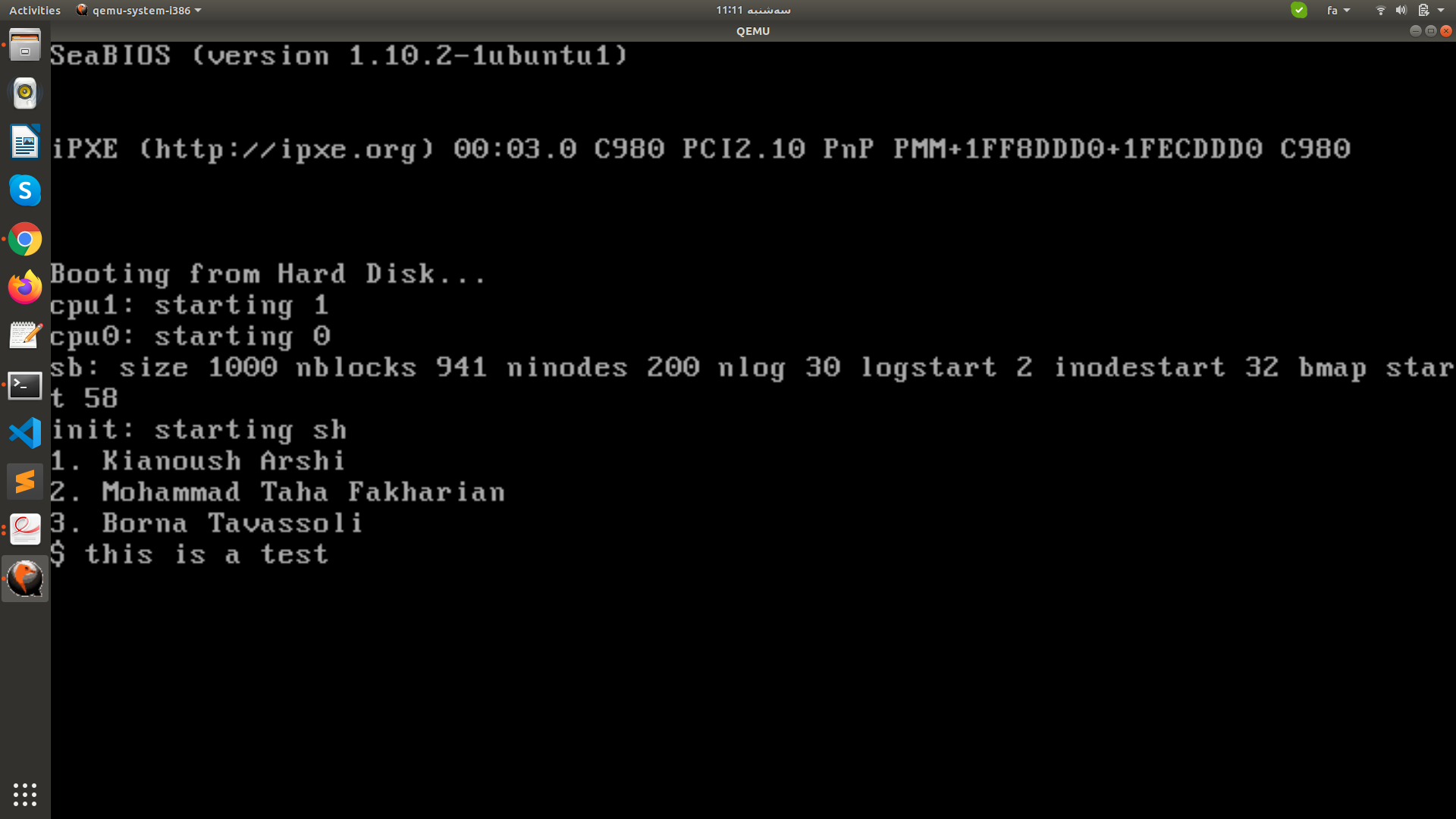


**۳. tab:**

متنی که داخل کنسول نوشته شده را کامل میکند :



Ctrl + A + “This “:



**اجرا و پیاده سازی یک برنامه سطح کاربر**

یک برنامه ساده با زبان C به نام prime نوشتیم که تمامی اعداد اول بین دو عددی که دادیم را در یک فایل به نام prime\_result.txt مینویسد.

کاربر با وارد کردن دستور <prime <number,number در کنسول یک پردازه جدید را شروع میکند و در نهایت با دستور cat میتواند خروجی را ببیند:



**5. سه وظیفه اصلی سیستم عامل را نام ببرید؟**

**یکی از وظایفش این است که واسطع است بین hardware و software.**

**برای لایه زیرین از مدیریت منابع استفاده میشود.**

**برای لایه بالاین از مدیریت کاربران و برنامه راهبردی**

**8.در MakeFile متغیرهایی به نام های UPROGS و ULIB تعریف شده است. کاربرد آن ها چیست؟**

**درxv6 اکثر اسم هایی که با u شروع میشود مرتبط با user هستند.متغیر UPROGSبرنامه های USER هستند و مثلا هنگامی که ما برنامه ی Icm را نوشتیم باید فایل \_lcm را به آن اضافه کنیم.متغیر ULIB لیستی از کتابخانه هایی که user از آنها استفاده می کند است.مانند malloc**

**11. برنامه ھای کامپایل شده در قالب فایل ھای باینری نگھداری میشوند. فایل مربوط به بوت نیز باینری است. نوع این فایل باینری چیست؟ تفاوت این نوع فایل باینری با دیگر فایلھای باینری کد xv6 چیست؟ چرا از این نوع فایل باینری استفاده شده است؟ این فایل را به زبان قابل فھم انسان (اسمبلی) تبدیل نمایید.**

**12. علت استفاده از دستور objcopy در حین اجرای عملیات make چیست؟**

**دستور objcopy کارش تبدیل فرمت و کپی کردن است.دو آبجکت فایل یکی را در انتهای دیگری کپی می کند.و فایل های خروجی را لودر در حافظه می گذارد.**

**14. یک ثبات عام منظوره ، یک ثبات قطعه ، یک ثبات وضعیت و یک ثبات کنترلی در معماری x86 را نام برده و وظیفه هر یکرا به طور مختصر توضیح دهید.**

**ثبات عام منظوره : همانطور که در کتاب هم اشاره شده، xv6 دارای ٨ ثبات عام منظوره میباشد. این ثبات ها عبارتند از e حرف eip نام به program counter یک و %esp و %ebpو %esiو %edi و %edx و %ecx و %ebx و %eax در این ثبات ها نشان دهنده extended است زیرا 32 بیت است. این ثبات ها نگهداری بعضی اشاره گرها، داده و برای نگهداری عملیات های ریاضی استفاده میشوند.**

**ثبات قطعه : آدرس استک، کد و داده در این ثباتها نگهداری میشود. برای مثال SS پوینتر به استک و CS پوینتر به کد و DS پوینتر به داده را نگه میدارد.**

**ثبات وضعیت: شامل اطلاعات راجع به وضعیت پردازنده است. EFLAGS در این بخش محسوب میشود و اطلاعات فلگهاییی نظیر carry و sign و zero را مشخص میکند.**

**ثبات کنترلی: کنترل CPU یا دستگاههای دیجیتال دیگر را در دست دارد. %cr0 و %cr2 و %cr3 و %cr4 از این ثباتها هستند. این ثباتها وظیفه تغییر مدل آدرسدهی، کنترل interrupt کنترل paging وهم پردازنده ها را دارند.**

**18. کد معادل entry.S در هسته لینوکس را بیابید .**

**در فایل های linux همان entry.S نام دارد**

**linux/arch/i386/kernel/entry.S**

**19. چرا این آدرس فیزیکی است؟**

**این مقدار باید یک آدرس فیزیکی باشد زیرا قسمت paging hardware نمی داند چگونه آدرس مجازی را ترجمه کند و هنوز page table ای ندارد که بخواهد توسط آن آدرس مجازی را ترجمه کند.**

**22. علاوه بر صفحه بندی در حد ابتدایی از قطعه بندی به منظور حفاظت هسته استفاده خواهد شد. این عملیات توسط seginit() انجام می گردد. همان طور که ذکر شد، ترجمه قطعه تأثیری بر ترجمه آدرس منطقی نمی گذارد. زیرا تمامی قطعه ها اعم از کد و داده روی یکدیگر می افتند. با این حال برای کد و داده های سطح کاربر پرچم SEG\_USER تنظیم شده است. چرا؟**

**تابع seginit در شروع هر پردازنده، یکبار اجرا می شود. همچنین این تابع با توجه به مقادیر SEG\_UCODE و SEG\_UDATA که در mmu.h به مقادیر 3 و 4 دیفاین شده اند. هیچ code**

**descriptor را بین هسته و کاربر به اشتراک نمی گذارد. با توجه به اولویت بیشتر برنامه های هسته، پردازنده جلوی وقفه را در حالتی که دسترسی از CLP=0 تا DLP=3 را می گیرد بنابراین در سیستم عامل xv6 هنگامی که پردازنده در حالت انجام برنامه سطح کاربر است، آدرس های بخش هسته غیرقابل دسترسی هستند. به همین دلیل به یک پرچم نیاز داریم که سطوح دسترسی بین هسته و کاربر را مشخص کند.**

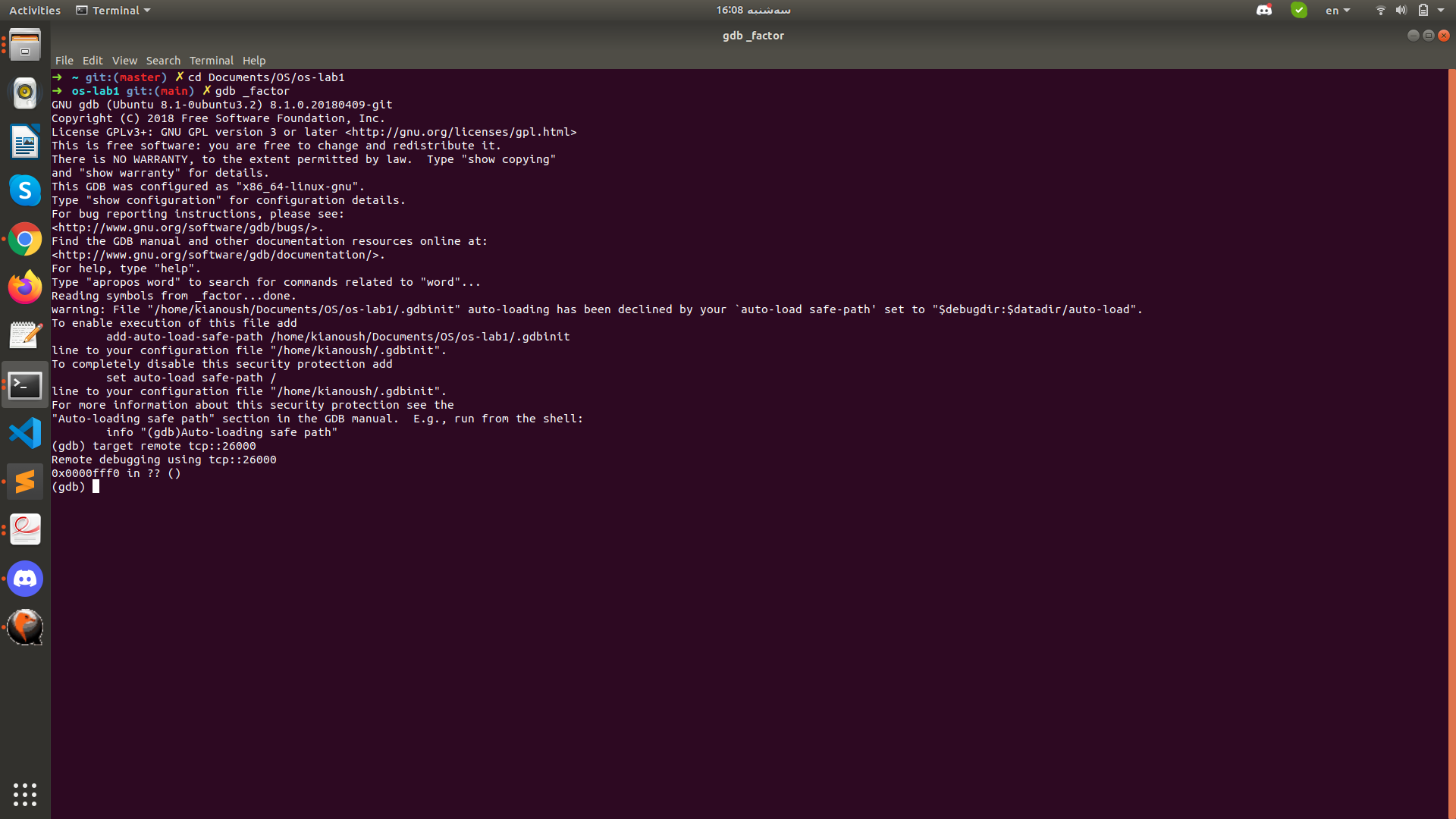
**23. جهت نگه داری اطلاعات مدیریتی برنامه های سطح کاربر ساختاری تحت عنوان struct proc ارائه شده است. اجزای آن را توضیح داده و ساختار معادل آن در سیستم عامل لینوکس را بیابید.**

**سایز حافظه ی پراسس-ادرس جدول صفحه- محل پایین استک کرنل بخش مربوط به این پراسس- وضعیت- آیدی- آدرس پراسس پدر- پنجره ی ترپ سیستم کال فعلی - چن- این که آیا پراسس کیل شده یا نه- فایل هایی که این پراسس باز کرده- دیرکتوری فعلی فایل- اسم پروسس )برای دیباگ کردن(.**

**27. کدام بخش از آماده سازی سیستم، بین تمامی ھسته ھای پردازنده مشترک و کدام بخش اختصاصی**

**است؟ زمان بند روی کدام ھسته اجرا میشود؟**

**اشکال زدایی**

****

**روند اجرای GDB**

**۱. برای مشاهده Breakpoint ها از چه دستوری استفاده می شود؟**

از دستور info breakpoints استفاده می شود.

**۲. برای حذف یک Breakpoint از چه دستوری و چگونه استفاده می شود؟**

می توان با استفاده از دستور

clear <filename>:<line>

می توان شماره خط را به gdb داد تا اگر Breakpoint ای که در آن خط وجود داشته باشد، آن را حذف کند.

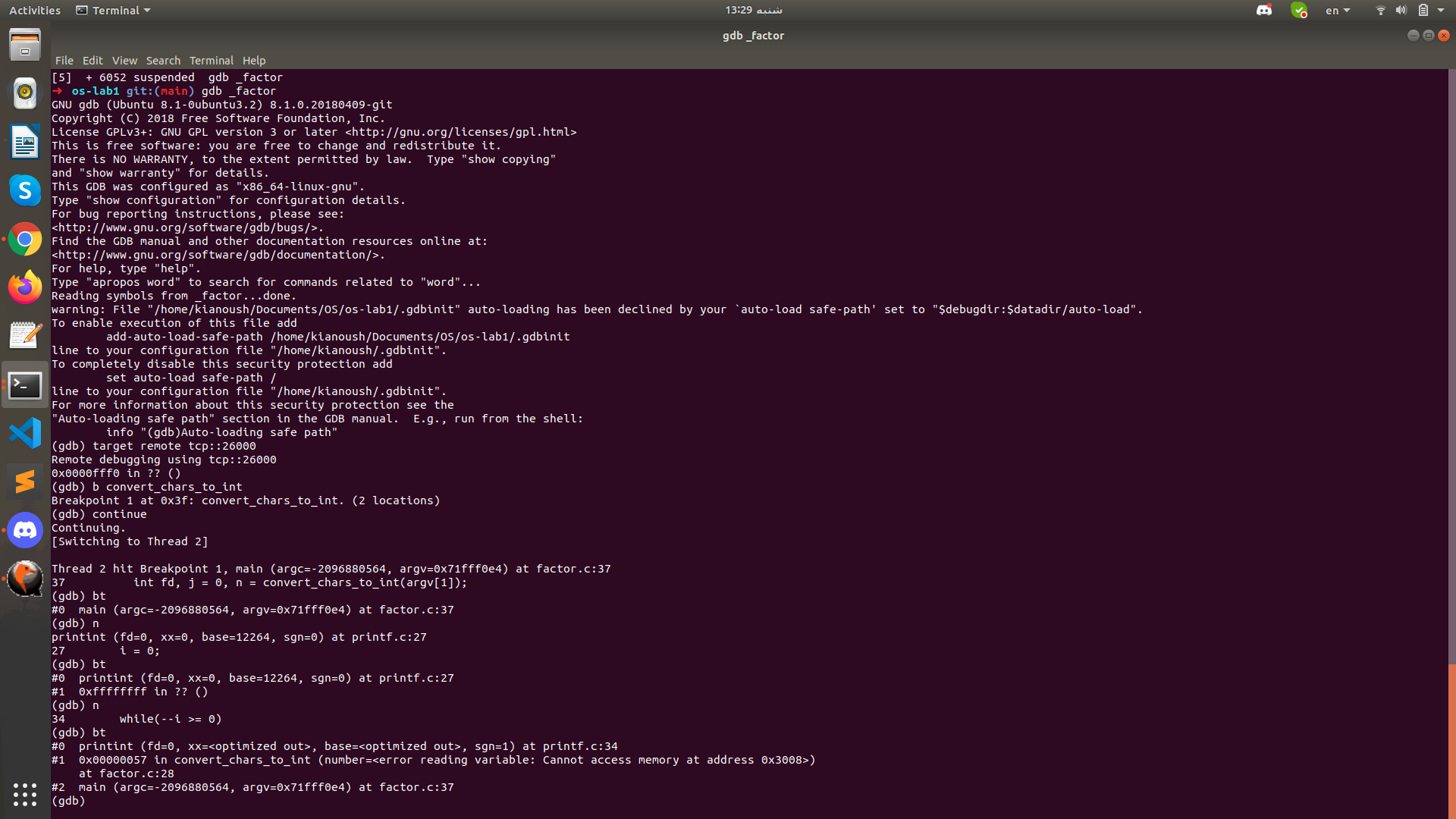
همچنین میتوان از دستور

**۳. دستور زیر را اجرا کنید. خروجی آن چه چیزی را نشان میدهد؟**

**$ bt**

این دستور مخفف backtrace هست و چیزی که نشان میدهد توابع فراخوانده شده و لود شده در استک میباشد.

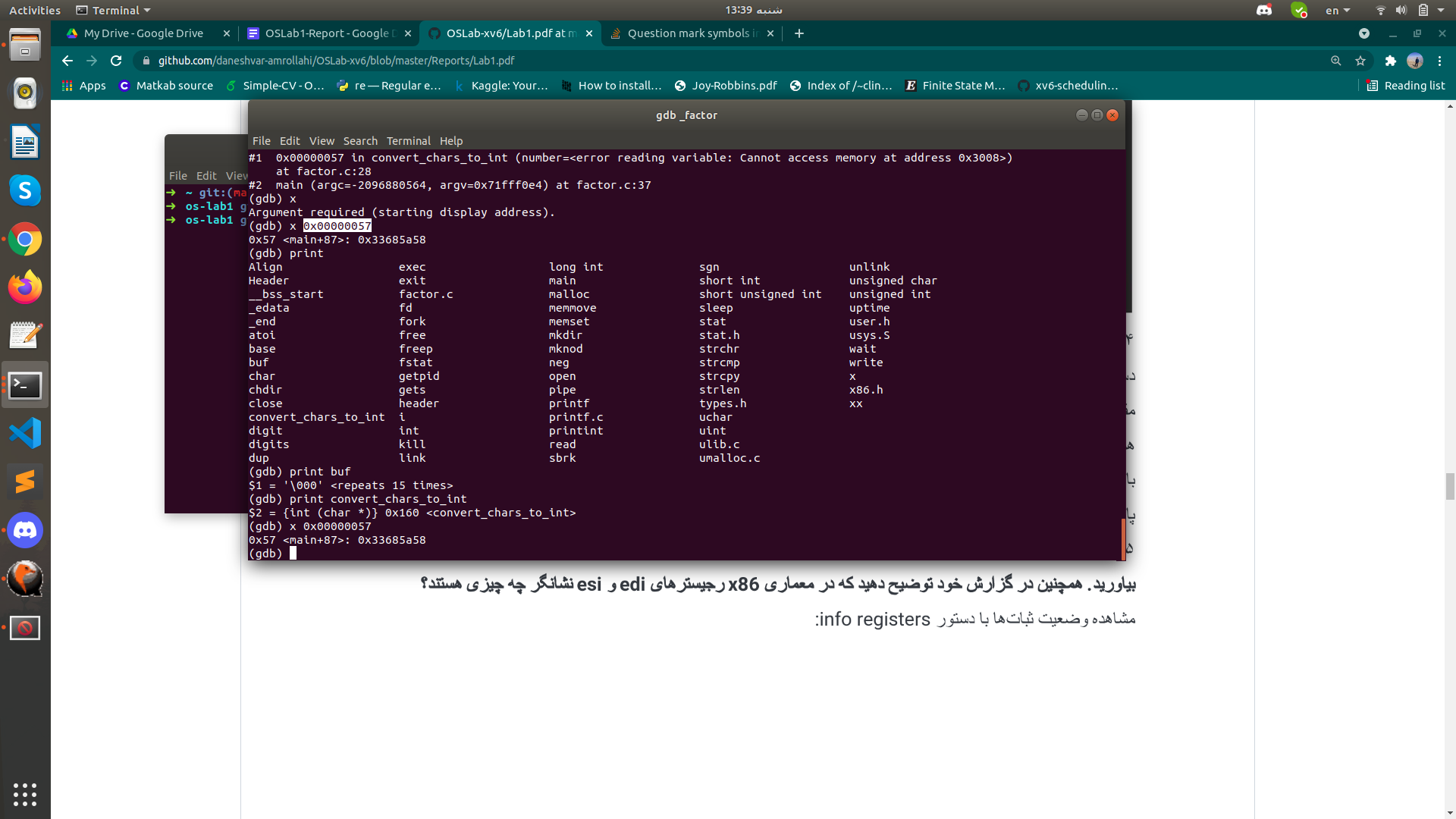
خروجی دستور به اینصورت است که در هر خط تابع فراخوانی شده را می نویسد و در خط بعد تابعی که تابع خط فعلی را فراخوانی کرده چاپ میکند.



**۴. دو تفاوت دستور x و print را توضیح دهید. چگونه میتوان محتوای یک ثبات خاص را چاپ کرد؟**

یکی از تفاوت های این دو دستور نحوه نمایش اطلاعات است.

تفاوت دیگری که دارند این است که دستور print با دریافت یک expression مقدارش را نشان میدهد اما دستور x مقدار ذخیره شده در یک آدرس را نمایش میدهد.

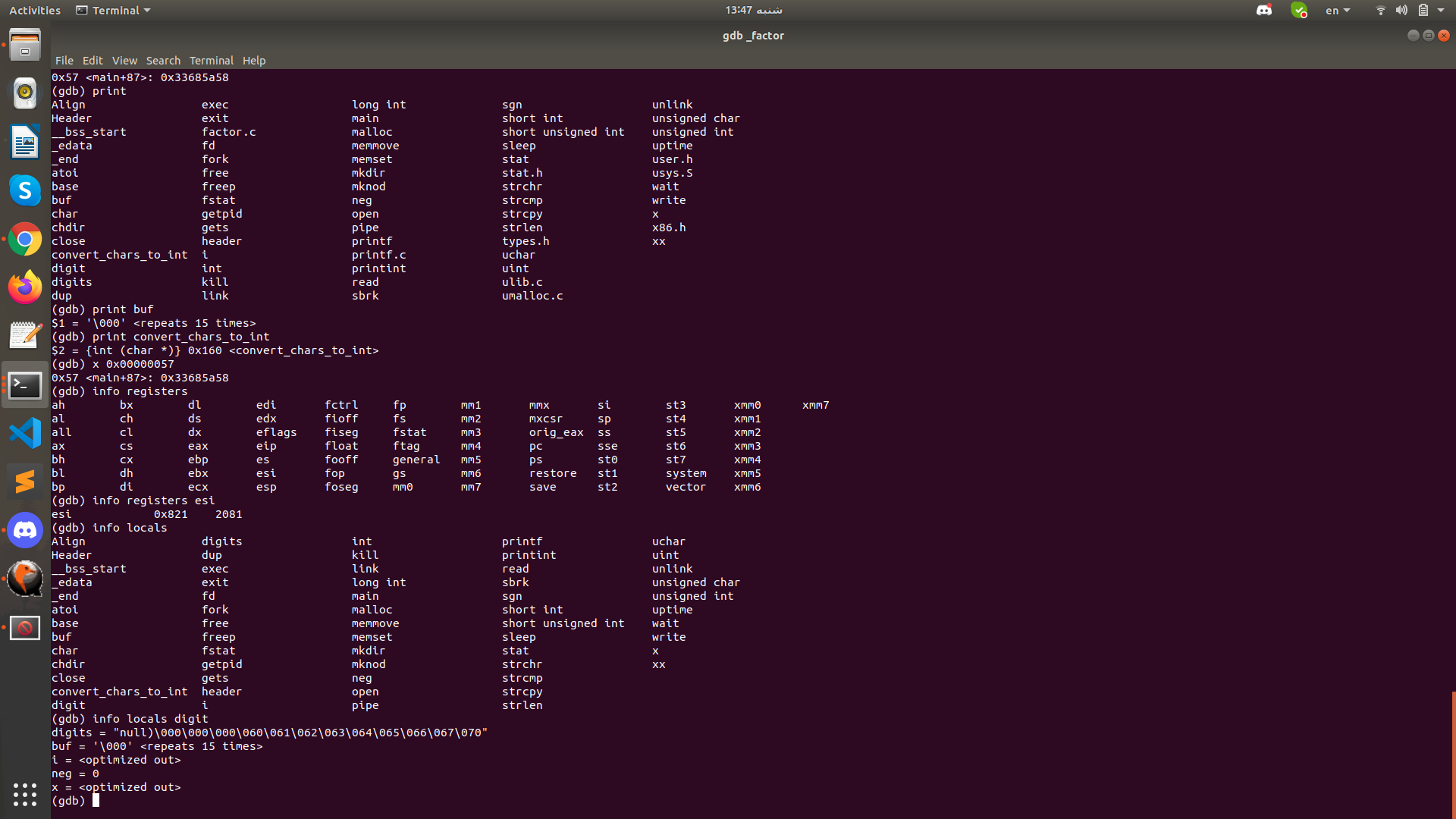


برای دریافت محتوای یک ثبات خاص میتوان از دستور< info register < register name استفاده کرد.

**۵. برای نمایش وضعیت ثباتها از چه دستوری استفاده میشود؟ متغیرهای محلی چطور؟ نتیجه این دستور را در گزارشکار خود بیاورید. همچنین در گزارش خود توضیح دهید که در معماری x86 رجیسترهای edi و esi نشانگر چه چیزی هستند؟**

برای وضعیت ثبات ها از info registers استفاده میشود.

برای متغیرهای محلی از info locals استفاده میشود.



دو رجیستر edi و esi درواقع به ترتیب به عنوان destination و source برای انجام یکسری از عملیات استفاده میشوند. معمولا به عنوان counter استفاده می شوند چون این دو ثبات با اجرای کد C تغییر نمیکنند.

**۶. به کمک استفاده از GDB، درباره ساختار input struct موارد زیر را توضیح دهید: توضیح کلی این struct و متغیرهای درونی آن و نقش آنها، نحوه و زمان تغییر مقدار متغیرهای درونی(برای مثال، input.e در چه حالتی تغییر میکند و چه مقداری میگیرد)**

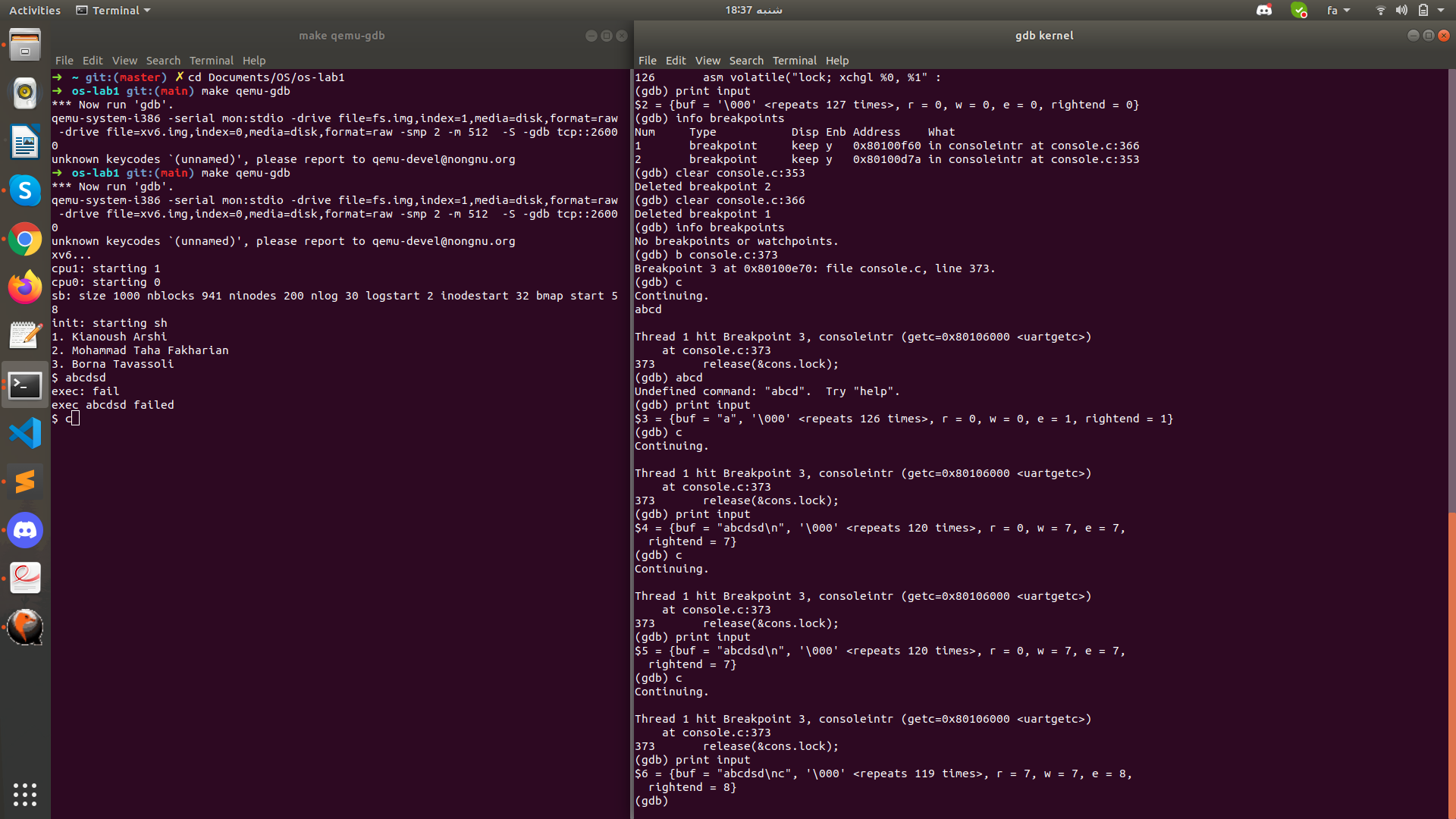
این استراکت دارای سه متغیر r,w,e میباشد.

r اندیسی از بافر را نشان میدهد که تا آنجا محتوای بافر خوانده شده و توسط سیستم عامل به آن محتوا رسیدگی شده. w اندیسی از بافر را نشان میدهد که تا آنجا در بافر نوشته شده و پس از اتمام یک خط در کنسول، این مقدار توسط e آپدیت میشود. e هم اندیسی از بافر را نشان میدهد که در حال حاضر میتوان مقدارش را با keyboard input عوض کرد.

با فشردن کلمات کیبورد و یا با کمک قابلیت های پیاده شده میتوان مقدار e را تغییر داد.

هنگامی که بافر پر میشود (128تا کاراکتر روی بافر قرار داشته باشد) و یا enter فشرده میشود مقادیر w و r آپدیت میشوند و برابر e میشود.البته در کد ما این مقادیر با rightend عوض میشوند که بخاطر اضافه شدن دستورات جدید به ساختار اضافه شده است. همچنین این استراکت یک آرای 128تایی از کاراکتر نیز به نام buf دارد که همان بافری است که در خط فعلی ذخیره کرده ایم.

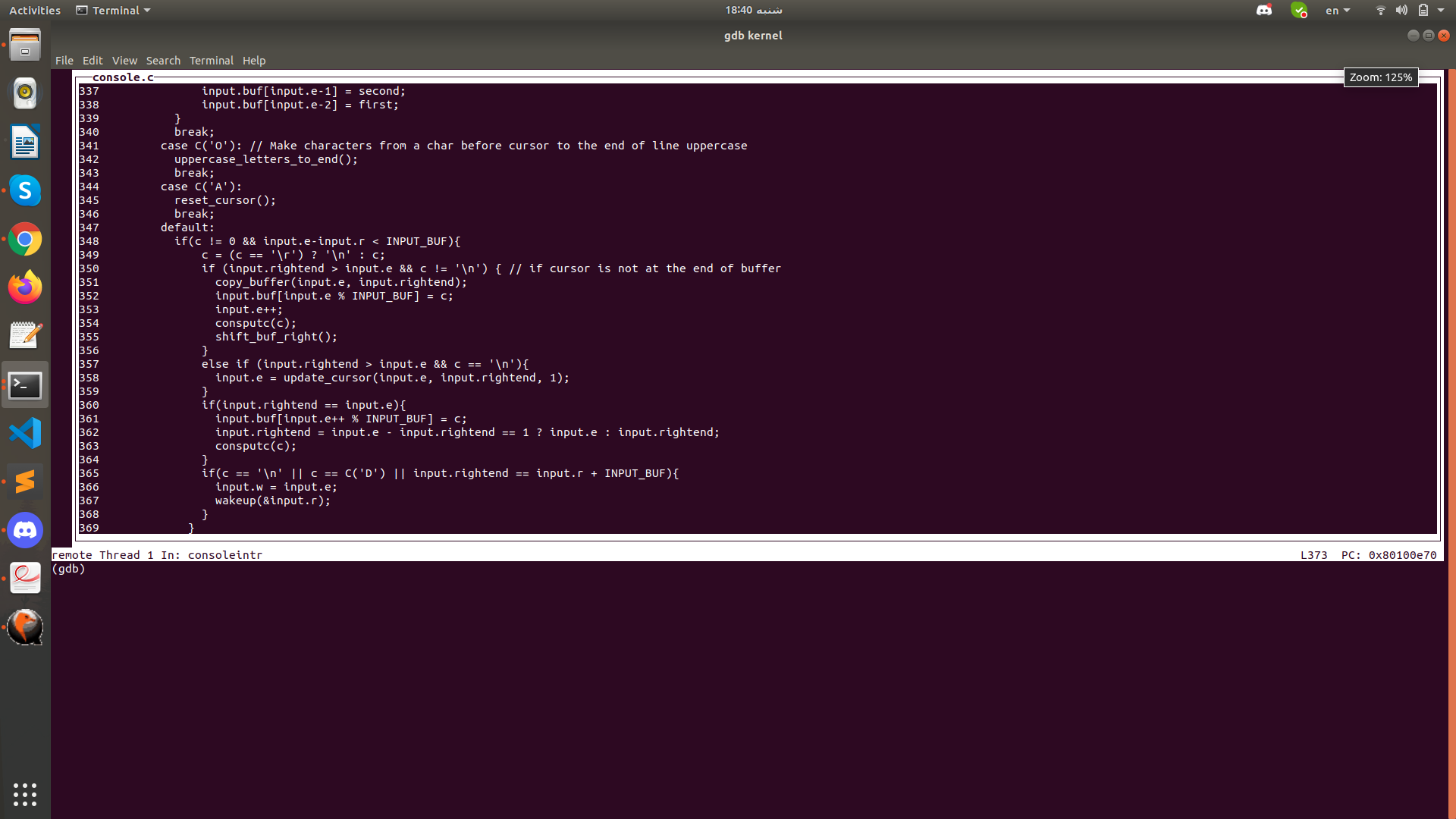
در ادامه به کمک GDB جاهایی که مقادیر input تغییر میکنند را مشاهده میکنیم:



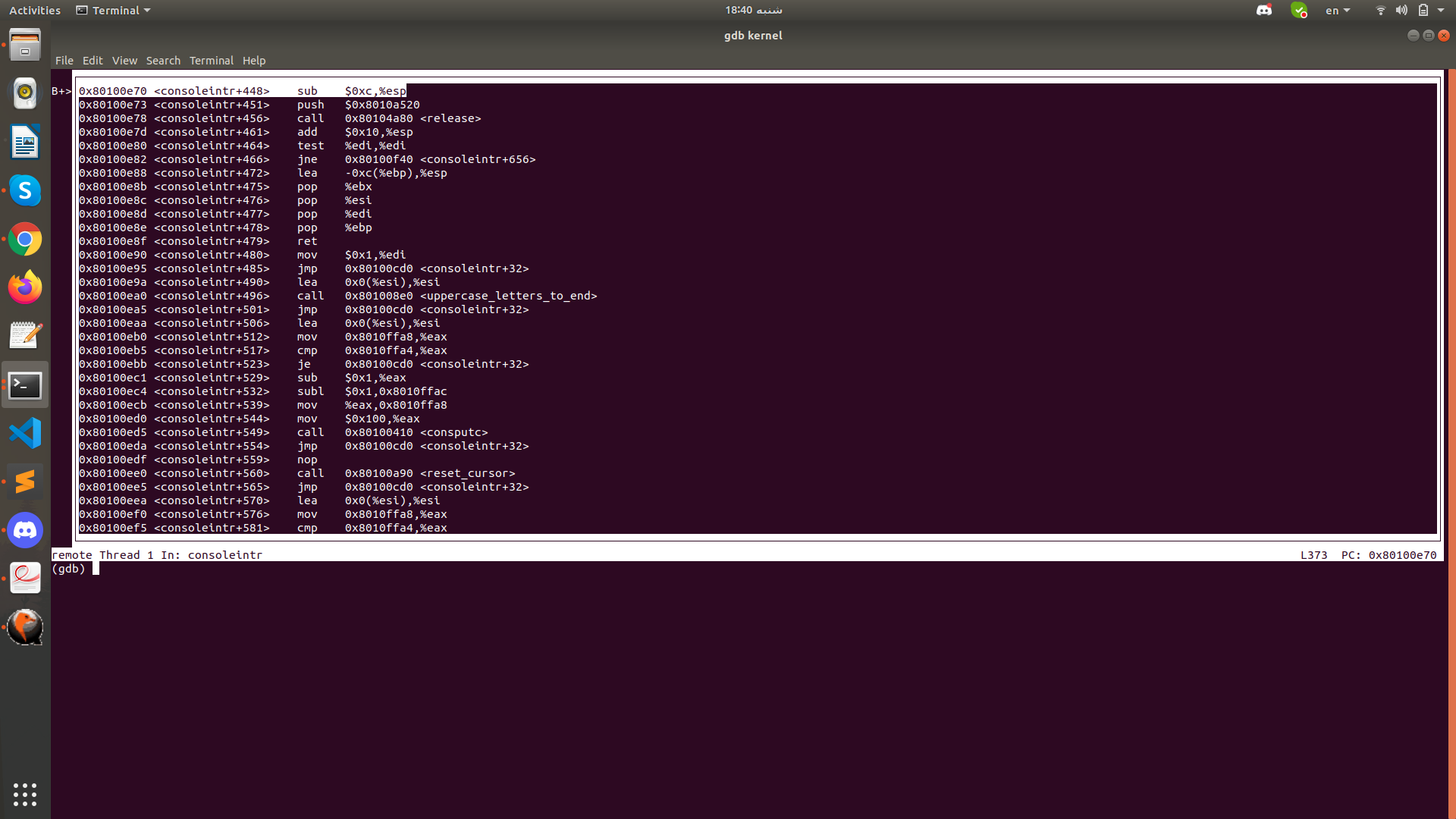
همانطور که مشاهده میشود با وارد کردن رشته abcdsd مقدار buf عوض میشود و با فشردن enter مقادیر r و w برابر اندیس اول خط دوم میشوند. مقدار e نیز با وارد کردن هر کاراکتر عوض میشود و درواقع نشانگر اندیسی هست که قرار است ورودی بعدی در آن خانه بافر قرار بگیرد.

**۷. خروجی دستورهای layout src و layout asm در TUI چیست؟**

در layout src کد سورس برنامه مشاهده میشود.



در layout sam کد برنامه به فرم اسمبلی نمایش داده میشود.



**۸. برای جابجایی میان توابع زنجیره فراخوانی جاری (نقطه توقف) از چه دستورهایی استفاده میشود؟**

up <number of jumps>:default = 1

down <number of jumps>:default = 1

