به فایلهایی که توسط یک پروسه باز شده میگن fd. از دید برنامه نویسی این یه عدد صحیح (معمولا بین 0 تا 1024) هست که فرخوان سیستمی ()open یا روالهای مشابه مثل fopen بر میگردونند. پروسه با این عدد که به یک فایل اشاره می‌کنه، میتونه توی اون فایل بخونه یا بنویسه. سه تا fd اول رزرو شده هستند. 0 برای ورودی استاندارد، 1 خروجی استاندارد، 2 خطای استاندارد. کاربر غیر برنامه نویس فقط با همین ۳ تا سرور کار داره. همینطور با ulimit هم میشه حداکثر تعداد fd که یک پروسه میتونه همزمان داشته باشه رو مشخص کرد. در ضمن این فایل میتونه یک فایل معمولی، یک سوکت و یا pipe باشه که هر کدوم روشهای مخصوص به خودشون رو برای خوندن و نوشتن دارن. برای فایلهای معمولی ()read و ()write، برای سوکت ها ()send و ()recv و برای pipe ها هم مثلا یه روش این هست که اول پروسه پدر ()fork میشه و یعدش ورودی یا خروجی استانداردش رو می‌بنده، و پروسه فرزند هم یکی از ورودی/خروجی هاش رو می‌بنده و از طریق ورودی/خروجی دیگه با هم ارتباط برقرار می کنند. امیدوارم این چند خط برای پیدا کردن چیزی که دنبالش بودید مفید باشه.

**بررسی مفهوم در لینوکس**

* File Descriptor چیست؟

در این مطلب می خواهیم در مورد مفهمومی به نام File Descriptor در لینوکس برای شما عزیزان صحبت کنیم. به زبان ساده File Descriptor شماره ای است که بصورت Unique یا منحصر بفرد به فایلی را که در سیستم عامل بصورت باز قرار دارد اختصاص داده می شود و بوسیله آن هر فایل از یکدیگر تمییز داده می شود. File Descriptor منابع اطلاعاتی و چگونگی دسترسی به آن را در سیستم عامل توصیف می کند. در سیستم عامل های لینوکس و همچنین یونیکس همه چیز یک فایل است.

فایل های معمولی ، دایرکتوری ها ، Symbolic link ها ، Local domain socket ها و حتی Device ها ( مثل Block device ها و Character device ها ) نیز از نگاه Kernel یا هسته سیستم عامل لینوکس یک فایل به حساب می آیند و File Descriptor های خود را دارند. حتی صفحه نمایش شما نیز یک فایل است و File Descriptor دارد . وقتی برنامه ای اجرا می شود خروجی برنامه به File Descriptor صفحه نمایش ارسال می شود و شما میتوانید خروجی را در صفحه نمایش سیستم تان ببینید. حال اگر خروجی به File Descriptor یک دستگاه پرینتر ارسال شود خروجی برنامه چیزی است که پرینتر آنرا چاپ می کند.

زمانی که شما یک برنامه یا دستور را اجرا می کنید سه فایل هستند که همیشه باز هستند ، این ها عبارتند از :

Standard Input

Standard Output

Standard Error

این سه فایل همیشه باز هستند زمانی که برنامه اجرا می شود. شماره مربوط به این فایل ها ( FD ) بصورت زیر هستند :

Standard Input STDIN 0

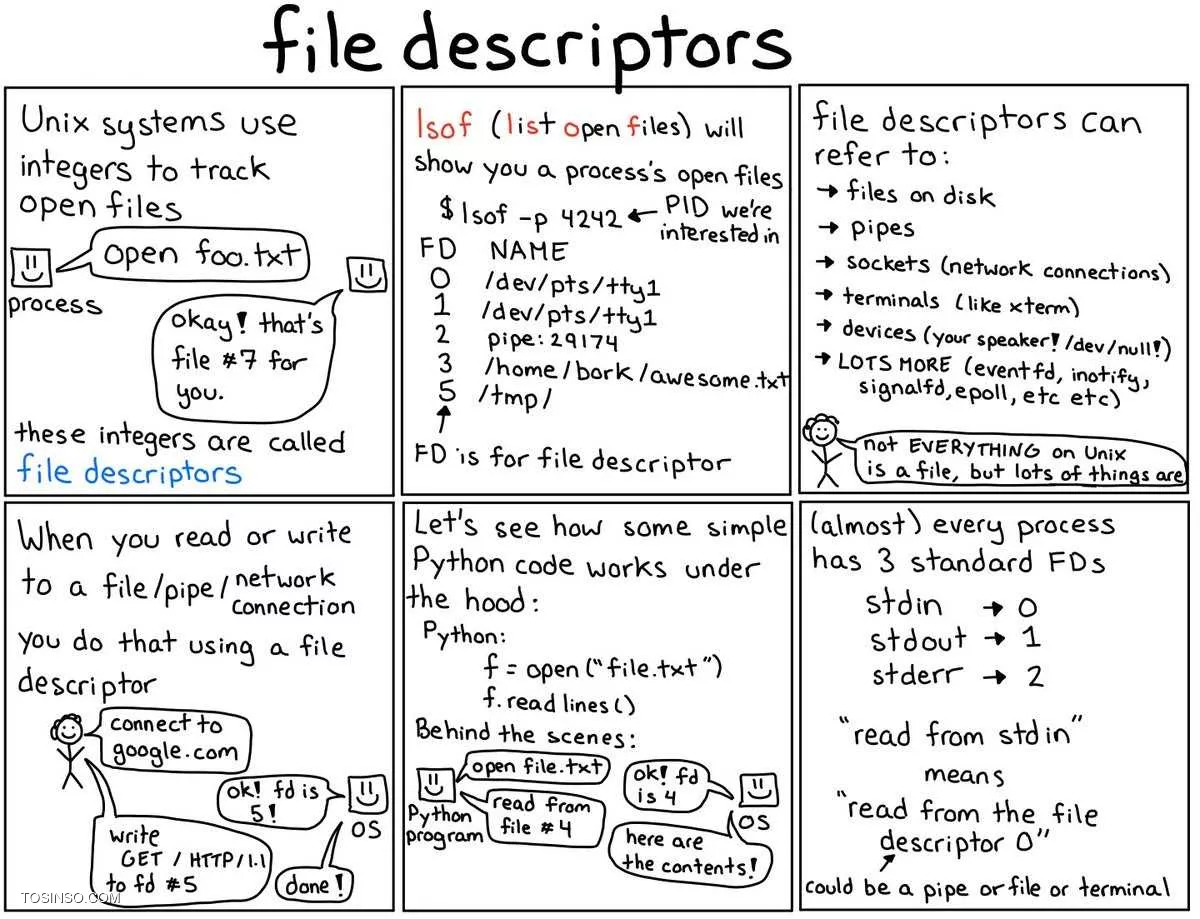
Standard Outpu t STDOUT 1

Standard Error STDERR 2

برای مثال زمانی که در جستجوی فایل هستید تعدادی از فایل ها معمولا با خطای permission denied و یا خطا هایی از این قبیل همراه هستند. این خطا ها را می توان در یک فایل ذخیره کرد ، بصورت زیر :

$ ls mydir 2>errorsfile.txt

File Descriptor برای Standard Error برابر 2 است. اگر هیچ دایرکتوری با نام mydir در سیستم وجود نداشت خروجی دستور در فایل errorfile.txt ذخیره خواهد شد. با استفاده از "<2" ما خطایی که در خروجی این دستور رخ خواهد داد را به فایل errorfile.txt در اصطلاح فنی Redirect می کنیم و شما در خروجی دستور در پنجره Terminal Emulator تان هیچ اروری را مشاهده نخواهید کرد.

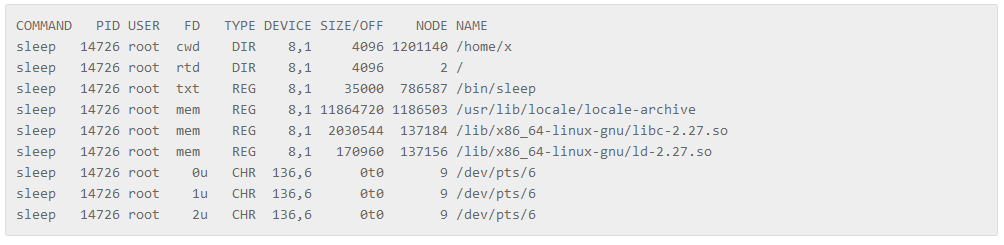


File Descriptor ها به PID ها Bound شده اند. به مثال زیر دقت کنید :

#>sleep 1000 &

[12] 14726

ما در دستور بالا یک پراسس با شماره 14726 ایجاد کردیم. با اجرای دستور lsof -p 14726 چیزی شبیه زیر را میتوانیم مشاهده کنیم :



چهارمین ستون یا FD که مخفف File Descriptor هست و ستون بعدی یعنی TYPE مربوط به File Descriptor و File Descriptor type هستند. همانطور که مشاهده می کنید برخی از مقادیر برای FD می تواند شامل مقدار cwd یا Current Working Directory ، مقدار txt یا Text file ، مقدار mem یا Memory mapped file و مقدار mmap یا Memory mapped device باشد. در سه سطر آخر کاراکتر u نشان دهنده این است که فایل هم برای Read و هم برای Write شدن باز شده است. بعبارتی یوزر می تواند روی آن دستگاه ( که حالا میتواند فلش درایو یا هر چیز دیگری باشد ) Read و Write داشته باشد. در ستون TYPE همانطور که از اسمش نیز مشخص است نشان دهنده نوع فایل است که REG یا Regular File به معنای فایل معمولی ( متنی ، صوتی ، تصویری ، ... ) است ، DIR یا Directory که خود یک نوع فایل است ، FIFO یا First In First Out و CHR یا Character Device را می توانیم مشاهده کنیم.

توجه کنید زمانیکه شما یک فایلی را باز می کنید یا یک فایل را ایجاد می کنید Kernel حداقل یک File Descriptor به Process آن اختصاص می دهد. و نیز زمانی که میخواهیم یک فایل را Read یا روی آن Write کنیم ما فایل را توسط File Descriptor شناسایی می کنیم که هنگام باز کردن یا ایجاد کردن به عنوان یک آرگومان برای Read یا Write کردن بازگشت داده می شود. نکته ای که در خصوص File Descriptor وجود دارد این است که عدد منفی و نیز اعشاری ندارد و تمامی File Descriptor ها اعداد مثبت را به خودشان اختصاص می دهند. نکته بعدی این است که Child Process های یک Parent Process یا پراسس والد File Descriptor هایشان را از Parent Process به ارث میبرند. امیدوارم مورد توجه شما قرار گرفته باشد.