03

Localizando arquivos com o locate

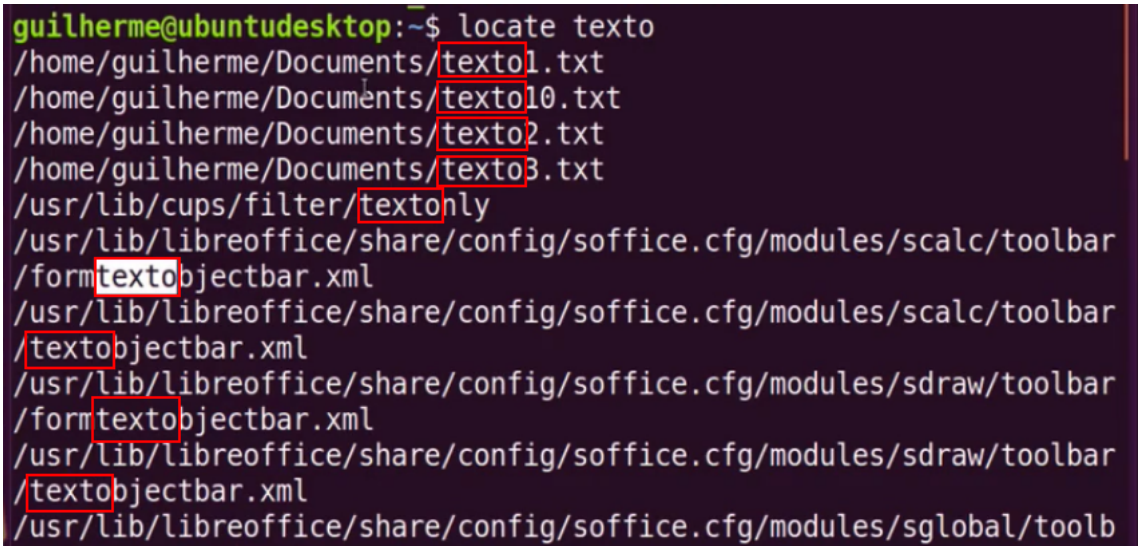
O locate e o updatedb

Um dos últimos pontos que a prova vai cobrar para nós nessa sessão é o `locate`, ou seja, como localizamos arquivos em nosso *Linux* a partir do Terminal. Temos diversas maneiras de localizar algo.

Vamos usar o comando `ls` para vermos os arquivos do diretório atual.

Veremos, mais adiante, outros comandos a medida que eles forem cobrados na prova, nesse instante nos interessa utilizar um para localizar arquivos em todo o nosso sistema de arquivos, portanti, usaremos o `locate`. O `locate` será usado para procurar arquivos que não temos nenhuma ideia de onde possam estar.

Imagine que, por exemplo, criamos alguns arquivos de texto e o nome desses arquivos eram `texto` e alguma coisa. Vamos digitar `locate texto` e dar um "Enter". Ele nos mostrará todos os arquivos que possuem a palavra "texto", observe:



Por exemplo, teremos o `texto1`, o `texto10`, o `textonly` e etc. Assim, todos os arquivos do `hd`, do sistema de arquivos, podendo ser remoto ou local, que possui a palavra `texto`, como o `textobjectar`, textonly e etc serão encontrados e mostrados. O `locate` é bom para realizar uma procura no sistema de arquivos inteiro. Repare como ele foi extremamente rápido ao fazer isso. O `locate`, por padrão, é bom para fazer buscas no sistema inteiro.

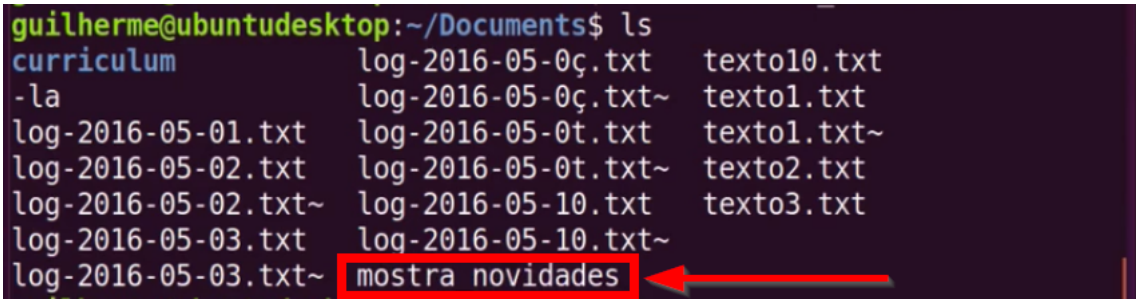
Vamos tentar alguma outra busca? Vamos digitar `locate mostra` e observar o que ele nos traz:

```
> locate mostra
/home/guilherme/mostra_idade
/home/guilherme/mostra_idade~
```

Repare que ele realiza uma busca muito rápida, ele procura em todo o sistema o `mostra` e trouxe o resultado muito rapidamente. Vamos testar e ver se é rápido mesmo? Vamos entrar no diretório `cd Documents/` e vamos criar um arquivo chamado `mostra_novidades`. Como queremos apenas criar um arquivo, queremos apenas tocá-lo e por isso digitamos `touch` seguido de um nome de um arquivo que não existe, mas que, após o "Enter" passará a existir. Em nosso caso colocaremos o nome de `mostra_novidades`:

```
> cd Documents/
~/Documents$ touch mostra_novidades
```

Para comprovar e ver se o arquivo foi criado podemos digitar `ls` e veremos que ele estará na lista:



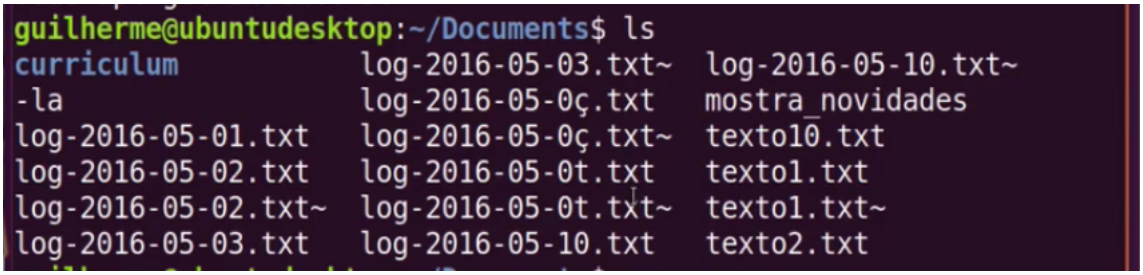
Para observarmos o que tem dentro desse arquivo podemos digitar `cat mostra_novidades` e veremos que ele não possui nada. Vamos sair desse diretório que acabamos de entrar digitando `cd ..` e vamos buscar novamente o `mostra`. Teremos o seguinte digitando `locate mostra`:

```
> locate mostra
/home/guilherme/mostra_idade
/home/guilherme/mostra_idade~
```

Repare que ele encontrou apenas dois `mostra`. Onde está o `mostra_novidades` que acabamos de criar?

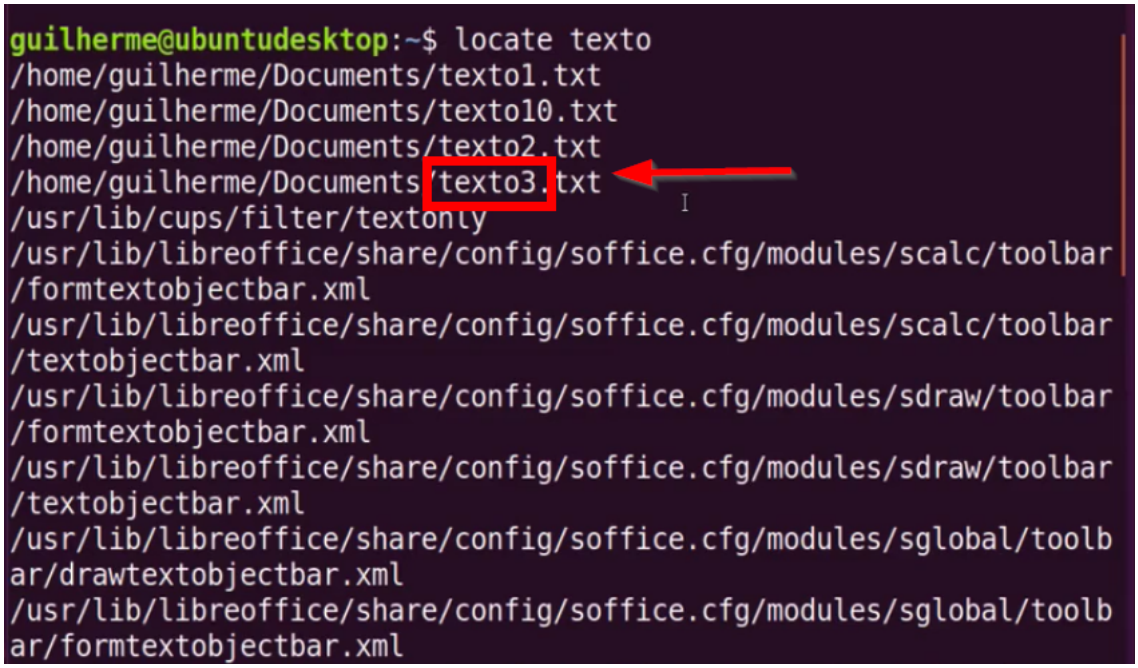
Ele encontrou apenas o `mostra_idade` e um arquivo de *backup* deste arquivo, que é o `mostra_idade~` criado após abrirmos no *gedit*. Mas, onde está o `mostra_novidades` que tínhamos criado?

Vamos entrar no *Documentos*, digitando `cd Documents` e vamos remover o arquivo `texto3` escrevendo `rm texto3.txt`. Vamos ver se ele ainda existe, digitando `ls`. Teremos o seguinte:



Bom, ele não consta mais em nosso `ls`.

Vamos voltar ao diretório anterior digitando `cd ..` e vamos escrever `locate texto`, teremos o seguinte:



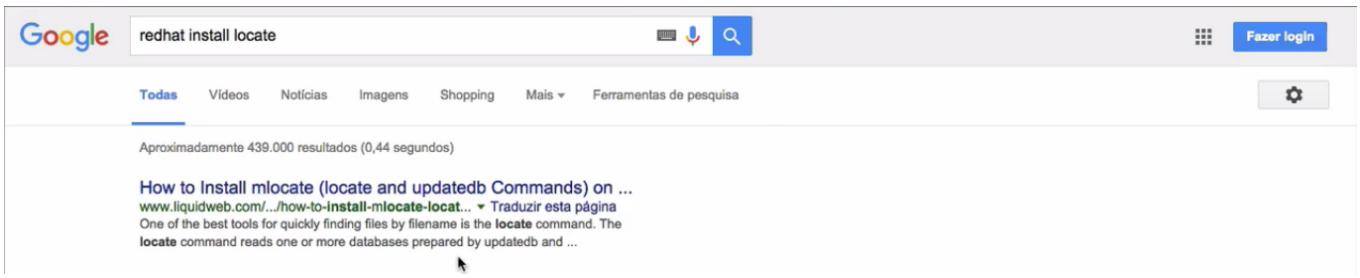
Veremos que o `texto3` ainda existe. O `locate` nos informa que o `texto3` segue existindo. O `texto3` na verdade não está mais aí, mas ele nos informa que ainda existe.

O que está acontecendo? Bom, lembra-se que o `locate` é extremamente rápido? Como ele consegue ser tão rápido?

O `locate` possui um banco de dados onde ele armazena todos os arquivos que existem, isto é, arquivos que ele é capaz de localizar. Então, isso é um banco de dados que ele mantêm e, por isso, toda vez que buscamos algo ele não necessita sair varrendo todos os arquivos e todos os diretórios para achar o que queremos. Ele vai rapidamente nesse banco de dados, que contem todos os nomes de diretórios e arquivos, encontra o que solicitamos e nos traz o que buscávamos. Esse é um processo que já está otimizado, como o `locate` é bastante utilizado no dia a dia ele simplesmente mantêm tudo já otimizado para realizar a busca.

Nem todas as instalações de *Linux* já vêm com o `locate` instalado. Em alguns *Linux* teremos que instalar o `locate` na mão. O `locate` em nosso *Ubuntu* já vêm instalado. Repare que se quisermos usar em algum outro sistema operacional, vai depender do nome do pacote que inclui o `locate` . No caso do *SUSE* vamos usar o `findutils-locate` e é esse pacote que vai ter o *locate*. Cada sistema operacional vai ter um pacote com um `locate` .

Então, teremos que procurar o que queremos, por exemplos, podemos querer o `redhat install locate` e veremos qual o `locate` que teremos que instalar. Se digitarmos no *Google* `redhat install locate` teremos o pacote `mlocate` que vêm com o `locate` para nós.

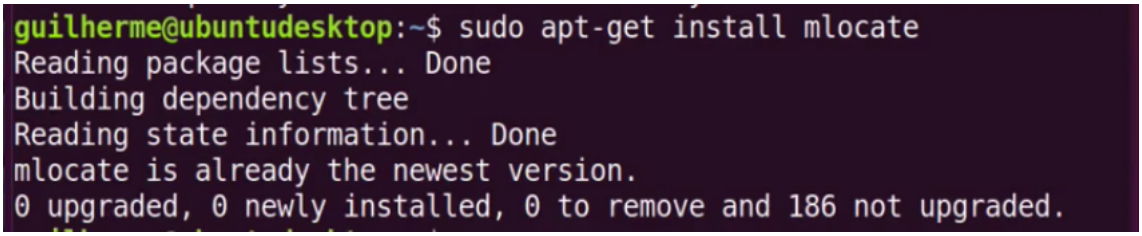


Em nosso `apt-cache search` se buscarmos pelo `findutils` teremos o seguinte:

```
> apt-cache search findutils
findutils - utilities for finding files--find, xargs mlocate -quickly find files on the filesystem '
```

Teremos o `mlocate` que é quem possui o `locate` para nós.

Podemos dgitar `sudo apt-get install mlocate` e teremos a seguinte resposta:



Temos que o `locate` já vêm instalado, já está com o banco de dados rapidão para buscarmos os nomes dos arquivos por isso, quando apagamos ou criamos um arquivo novo esse banco fica desatualizado e, por esta razão o `locate` não funciona como deveria.

Como fazemos para atualizar esse banco de dados? Rodamos um programa chamado `updatedb` .

Ao digitarmos o nome do programa teremos o seguinte:

```
> updatedb
updatedb: can not open a temporary file for '/var/lib/mlocate/mlocate.db'
```

O `updatedb` deve ser rodado como um superusuário, então, digitaremos `sudo updatedb` e damos um "Enter" e agora sim temos o banco de dados atualizado. Agora, se dermos um `locate mostra` teremos, por fim, o `mostra_novidades` :

```
> locate mostra
mostra_idade
mostra_idade~
mostra_novidade
```

Então, temos que ficar executando o `updatedb` toda vez para termos a certeza de que está atualizado? Isso até parece um pouco inútil. Bom, não necessariamente necessitamos fazer isso! Podemos rodar um `locate` com a opção `-e` que é de arquivos apagados e arquivos que não existem mais. Então, ele não vai mais nos mostrar arquivos que não existem mais. É uma maneira de não pegar os falsos positivos. Nesse caso, por exemplo, se digitamos o `locate -e texto` ele não nos mostrará mais aquele `texto3` que foi apagado e que antes foi nos mostrado:

```
guilherme@ubuntudesktop:~$ locate -e texto
/home/guilherme/Documents/texto1.txt
/home/guilherme/Documents/texto1.txt~
/home/guilherme/Documents/texto10.txt
/home/guilherme/Documents/texto2.txt
/usr/lib/cups/filter/textonly
/usr/lib/libreoffice/share/config/soffice.cfg/modules/scalc/toolbar
/formtextobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/scalc/toolbar
/textobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/sdraw/toolbar
/formtextobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/sdraw/toolbar
/textobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/sglobal/toolb
ar/drawtextobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/sglobal/toolb
ar/formtextobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/sglobal/toolb
ar/textobjectbar.xml
/usr/lib/libreoffice/share/config/soffice.cfg/modules/simpress/tool
bar/formtextobjectbar.xml
```

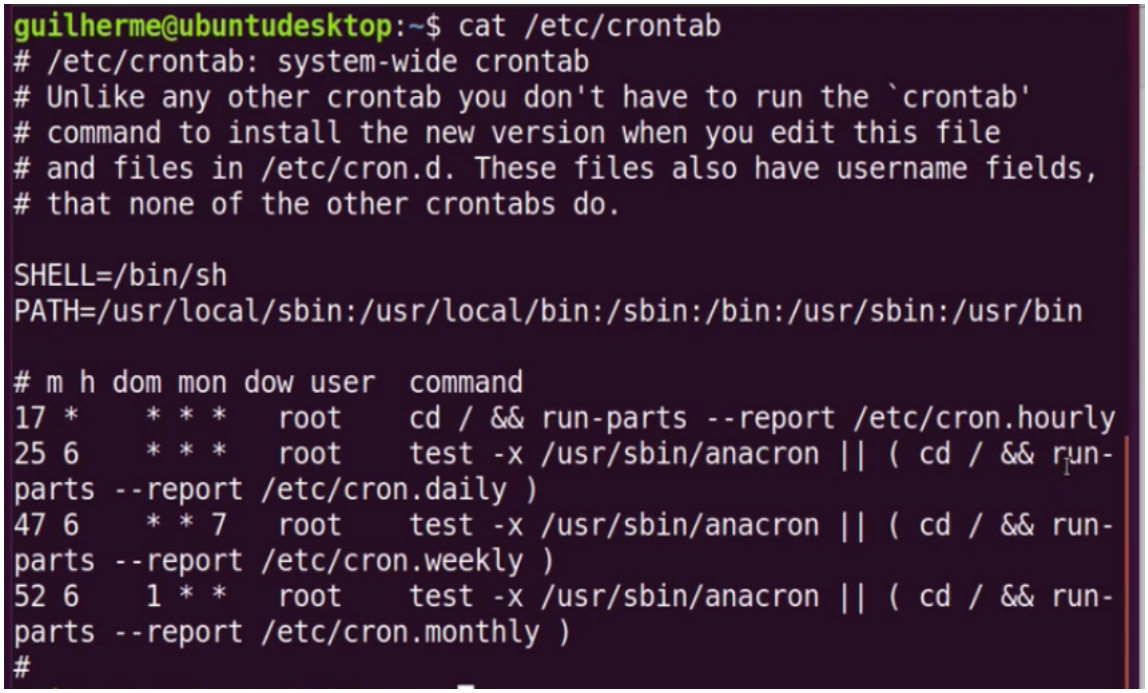
Vamos apagar o `texto2` para fazermos um teste. Digitaremos `rm Documents/texti2.txt` e rodaremos na sequência o `locate -e texto`. Vamos buscar o `texto2`:

```
guilherme@ubuntudesktop:~$ locate -e texto
/home/guilherme/Documents/texto1.txt
/home/guilherme/Documents/texto1.txt~
/home/guilherme/Documents/texto10.txt
```

Ele não nos mostra nada, ele usou o banco de dados desatualizado mas antes de nos mostrar algo ele verificou se aquele arquivo ainda existia ou não e foi o `-e` que fez isso. Ele só nos mostra o `texto 1` e o `texto 10`.

Mas se criarmos um arquivo, ele não terá como saber, e teremos que rodar o `updatedb`, não é mesmo? Verdade! Por isso que a maior parte das instalações de *Linux* que já instalam o `mlocate` para nós já configuram para ele ser rodado de tempos em tempos, com uma certa frequência, assim ele acaba sendo atualizado. O programa que faz isso é até um programa que citamos anteriormente, o `crontab`. A configuração dele em geral vai depender de cada instalação *Linux* e fica no

`/etc/crontab` . Vamos digitar `crotab /etc/crontab` e dar um "Enter". Teremos: r



Repare que é um arquivo genêrico e ele até existe no *Ubuntu*, mas se procurarmos não vamos achar a palavra `mlocate` , só vamos achar que todo o dia roda o `daily` e que toda semana roda o `weekly` e todo o mês roda o `mounthly` e toda hora roda o `hourly` .

Vamos dar um `clear` e digitar `ls /etc/cron` e damos um "Enter":

```
> ls /etc/cron
cron.d/         cron.hourly/        crontab
cron.daily/     cron.monthly/       cron.weekly/
```

Se digitarmos `ls /etc/cron.hourly/` não teremos nada. Vamos escrever `ls /etc/cron.daily/` e dar um "Enter". Teremos o seguinte:

```
> ls /etc/cron.daily/
0anacron        cracklib-runtime        passwd
apache2         dpkg                    popularity-contest
apport          logrotate               update-notifier-common
apt             man-db                  upstart
bsdmainutils    mlocate
```

E encontramos o `mlocate` . Vamos dar uma olhada nesse arquivo digitando `cat /etc/cron.daily/mlocate` e teremos o seguinte:

```
guilherme@ubuntudesktop:~$ cat /etc/cron.daily/mlocate
#! /bin/bash

set -e

[ -x /usr/bin/updatedb.mlocate ] || exit 0

if which on_ac_power >/dev/null 2>&1; then
    ON_BATTERY=0
    on_ac_power >/dev/null 2>&1 || ON_BATTERY=$?
    if [ "$ON_BATTERY" -eq 1 ]; then
        exit 0
    fi
fi

# See ionice(1)
if [ -x /usr/bin/ionice ] &&
    /usr/bin/ionice -c3 true 2>/dev/null; then
    IONICE="/usr/bin/ionice -c3"
fi

flock --nonblock /run/mlocate.daily.lock $IONICE /usr/bin/updatedb.
mlocate
```

Ele roda esse `script` uma vez por dia e ele chama o `updatedb` . Assim, ele roda o `updatedb` uma vez por dia, isto é, ele atualiza uma vez por dia o banco de dados. Temos, então, que ocorre uma atualização baseada no sistema de arquivos uma vez por dia, o que está "ok". Mas se não estiver "ok" para você, podemos digitar `sudo updatedb` ou podemos configurar o `crontab` . O `crontab` abordaremos mais para frente quando falarmos sobre tópicos mais avançados, pois, por enquanto não é nosso preocupação, para esta prova ao menos.

Precisamos do `-e` que é extremamente interessante, sabemos que temos que rodar o `updatedb` e que temos o `locate` e como o `locate` roda como superusuário, por padrão, ele busca todos os arquivos, inclusive, arquivos que nós não deveríamos ter acesso, mas que apenas o superusuário ou outro usuário tem.

Nos deparamos com um perigo, uma questão de segurança delicada. Para resolvermos esse problema de segurança, isto é, de algum usuário descobrir que existem arquivos que não era para ele descobrir que existiam, o `updatedb` tem duas maneiras de ser rodado, com o `root` ou com o `nobody` . O `updatedb` que atualiza e que busca as coisas pode ser rodado como um superusuário por padrão, mas na hora que ele vai atualizar, ele pode ser rodado como um usuário `root` ou como um usuário `nobody` , ninguém.

Se ele for `root` ele vai pegar tudo de todo mundo e colocar no banco e pronto, mas se ele for rodado no modo de usuário `nobody` , ele só vai conseguir acessar informações que esse usuário tem acesso. Ele muda completamente o uso dele. Em um caso ele consegue listar, armazenar, todos os diretórios e arquivos e no outro caso, apenas àquilo que o usuário possui acesso. Mais adiante, falaremos bastante sobre como dar e evocar acesso na parte de permissões.

Repare que se formos para a documentação do `updatedb` , digitando `man updatedb` e dermos um "Enter" encontraremos as suas diversas opções, inclusive , maneiras distintas de rodar ele. Teremos exemplos de como rodar ele para executá-lo acessando apenas um usuário, sem acessar outros usuários. Por padrão ele cria o banco de dados no `/var/lib/mlocate/mlocate.db` e ele usa a configuração do arquivo `/etc/updatedb.conf` .

Vamos ver esse arquivo? Para isso digitamos `less /etc/updatedb.conf` e dando um "Enter" teremos o seguinte:

Temos esse arquivo que nos mostra, simplesmente, algumas das opções que ele usa para o `updatedb` por padrão.

Repare a importância `updatedb` para nós no dia a dia, ele mantêm atualizado o sistema de busca do `locate`, do banco de dados, para que possamos usar o `locate` de maneira a ser extremamente rápido. Mas, ele entra em questões de segurança que ou o usuário `root` consegue ver tudo ou o que somente o usuário `nobody` pode ver por padrão.

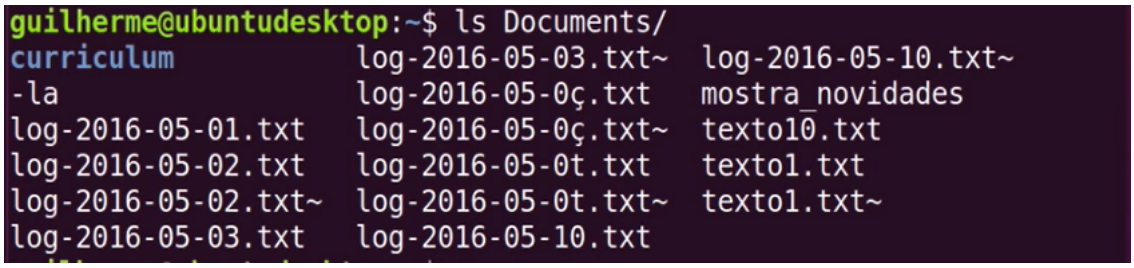
E podemos rodar isso de maneiras customizadas, por exemplo, podemos rodar só para o nosso usuário, só para outro usuário ou apenas para determinados usuários. Podemos configurar isso.

Na prova ele apenas quer que saibamos para que servem: o `locate`, o `updatedb`, o `-e`. O `locate` é usado para buscar, o `updatedb` é utilizado para atualizar o banco de dados, o `-e` é usado para tirar os arquivos que já não fazem mais sentido, ou seja, as referências à diretórios que não tem sentido, pois, foram removidos. O `updatedb` roda como `sudo`. E `mlocate` é o nome do pacote que instalamos no *Linux* pois é um *Ubuntu Debian*, se fosse um *SUSE*, seria outro pacote, cada um possui suas especificidades, isto é, pacotes distintos a serem atualizados.

Padrões de `locate`

Por fim, vamos mostrar que outro tipo de padrão podemos utilizar para procurar arquivos.

Repare que no `Documents/` temos diversas coisas, para verificar isso basta digitar `ls Documents/`:



Temos alguns arquivos que se chamam `log` alguma coisa. Lembra-se que aprendemos o `globbing` no `ls` e que ele nos permitia utilizar o "*", "[]", "?". Adivinhe! O `locate` também permite esse tipo de coisa. Então, podemos digitar `locate log` que estaremos dizendo para ele localizar tudo o que contêm a palavra "log".

Isso quer dizer que ele procura a palavra "log" em qualquer lugar do `path`, seja no nome do arquivo, seja no diretório onde ele está, isto é, em qualquer lugar! No momento em que colocamos um carácter que é curinga, isto é, especial, ele deixa de procurar em qualquer lugar. Por exemplo, se digitamos `locate log*` ele passa a procurar o que começa com `log` e termina com qualquer coisa. Dando um "Enter" nisso teremos o seguinte:

```
> locate log*
/home/guilherme/log completo. txt
/home/guilherme/log completo. txt~
```

Repare que fica bem estranho! Pois, os arquivos do diretório "Documents" também começam com `log` e ele não nos traz isso.

O que aconteceu?

Já vimos sobre o *globbing*, então, vamos retomar um pouco sobre como ele funciona. Se não escapamos, o asterisco, é interpretado pelo *bash* e no diretório atual temos dois arquivos que chamam-se `log` . Vejamos:

```
> ls
arquivos.zip            falha~                mostra_idade~         temp
Desktop                 help                  Music                 Templates
Downloads               log completo.txt      Pictures              Videos
examples.desktop        loja                  sucesso
falha                   mostra_idade          sucesso~
```

E como ele traduziu isso? Traduziu como `log\ completo.txt` e `log completo. txt~` , então, o comando que ele executou foi `locate log\ completo.txt log\ completo.txt~` e, por isso, nos trouxe apenas esses dois `log` . É como se tivéssemos isso:

```
> locate log\ completo.txt log\ completo.txt~
/home/guilherme/log completo.txt
/home/guilherme/log completo.txt~
```

Temos que ter muito cuidado quando queremos fazer o `locate` . Na verdade, não queremos fazer um `locate log*` , o que queremos é um `locate "log*"` . Se digitarmos isso ele não nos trará nada. E por que isso acontece? No momento em que colocamos o curinga ele procura exatamente o que queríamos. Antes, ele meio que colocava um curinga antes e um no final. Agora não, agora tem que começar com "l".

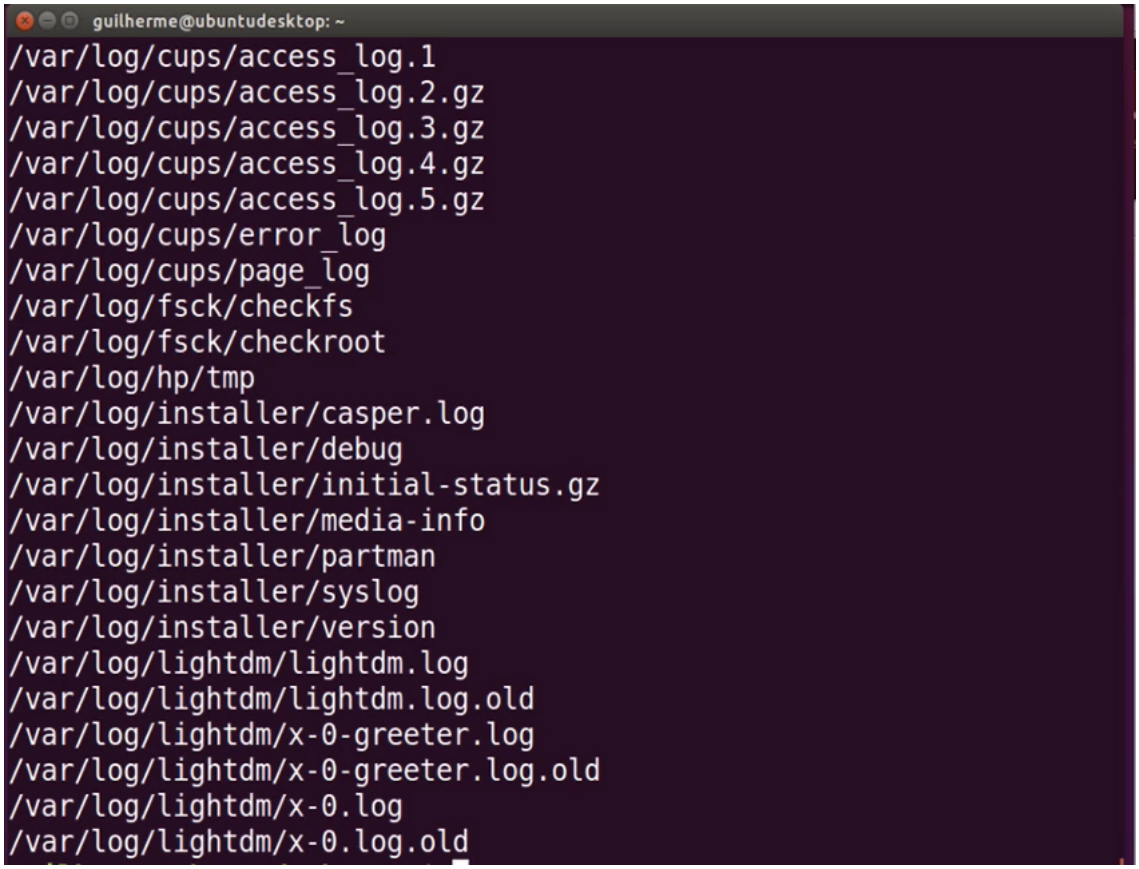
Mas, você pode estar pensando que temos diversos arquivos que começam com "l" no diretório. Vamos digitar `ls` e ver o que acontece:

```
> ls
arquivos.zip            falha~                mostra_idade          temp
Desktop                 help                  Music                 Templates
Documents               log completo.txt      Pictures              Videos
Downloads               log completo.txt~     Public                zip
examples.desktop        mostra_idade          sucesso
falha                   mostra_idade          sucesso~
```

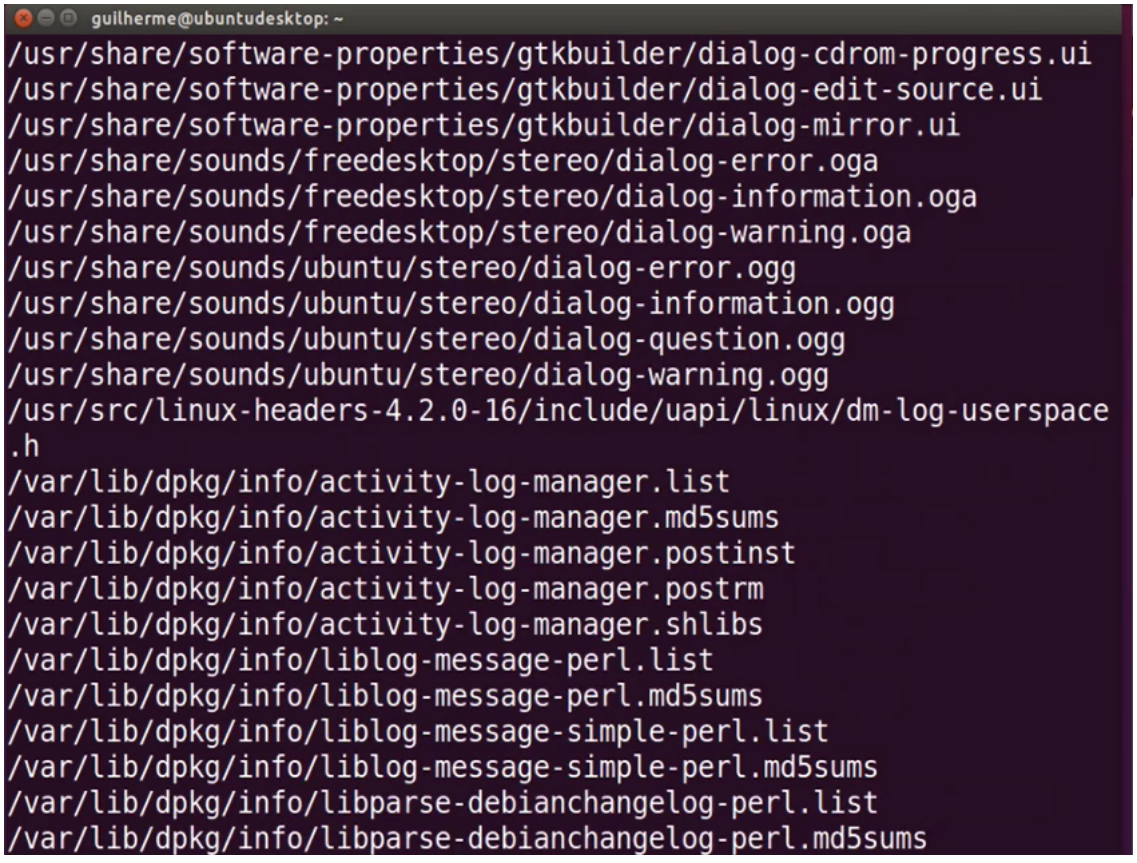
Na verdade, eles não começam com "l", eles iniciam com `/home/guilherme` , para confirmar isso basta digitar `pwd` :

```
> pwd
/home/guilherme
```

Repare que o `locate` , quando usamos um curinga, teremos que usar eles antes, por exemplo, digitaremos `locate "*log*"` , quando queremos falar "log" + qualquer coisa e, então, ele nos trará "qualquer coisa":

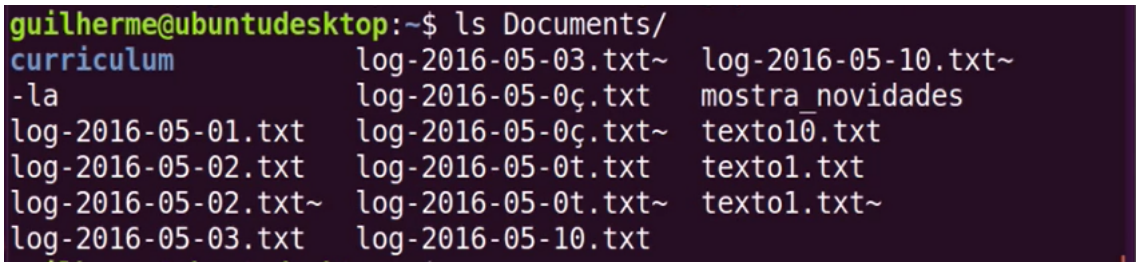


Aí ele nos mostra qualquer coisa que possua "log" em qualquer lugar. E se quisermos, na verdade, aquilo que está apenas no diretório "Documents" e que possui um hífen? Digitaremos `locate "*log-*"` e teremos o seguinte:

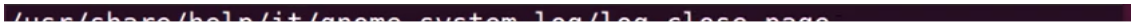


Teremos tudo que é `log` seguido de um hífen.

Vamos observar, agora, o `Documents` . Para isso, basta digitarmos `ls Documents/` , teremos o seguinte:



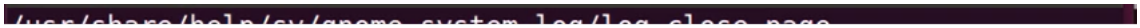
Agora, queremos buscar algo que comece com "log" e é seguido de hífen. Queremos tudo o que está em qualquer diretório e que o modo do arquivo comece como `log` e seja seguido de um sinal de "menos" e qualquer coisa. Para isso, digitaremos `locate "*/log-*"` , teremos o seguinte:



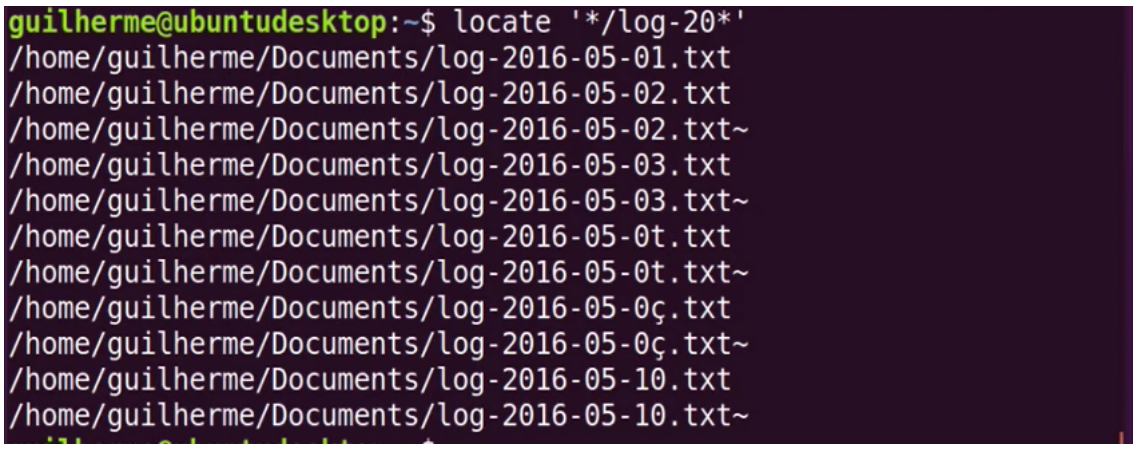
Repare que ele trouxe para nós os arquivos que queríamos, mas ele trouxe também outras coisas, arquivos que ainda são `log-` alguma coisa. Observe:

Repare que, temos o asterisco, temos o ponto de interrogação e os colchetes. Temos esses três caracteres em nosso `locate` e como eles funcionam mesmo? Eles funcionam, por padrão, ao procurarmos a palavra solta ela é buscada em qualquer lugar, mas no momento em que colocamos um asterisco, precisamos tomar cuidado, pois ele fará *globbing*.

O que vamos querer fazer? Provavelmente, vamos querer rodar com aspas. Só que, ele começa a buscar algo que começa com "l" e nada começa com "l". Por isso, se digitarmos `locate 'log*'` não teremos nenhuma resposta. Na verdade, tudo inicia com `/`. Então, temos que modificar isso, vamos digitar `locate '*/log*'` e então, ele nos trará o que buscávamos:



E se o que queremos for escrito com hífen? Vamos digitar `locate '*/log-*'` e ele nos trará o que estamos buscando. E se quisemos um "log" que seja seguido do número 20? Vamos escrever `locate '*/log-20*'` e ele nos mostrará somente aquilo que estiver nesse conforme:



Não necessariamente precisaria ser o ano que acompanha o `log` , poderia ser qualquer coisa que tenha o número vinte.

Repare como funciona o `locate` , podemos utilizá-lo com nossos padrões, o asterisco, o colchete e o ponto de interrogação. A dica é trabalhar com eles, procurar diversos arquivos e brincar bastante.

Vamos ver um último caso!

Lembra que quando fizemos o `locate log*` ? Tínhamos o seguinte:

```
> locate log*
/home/guilherme/log/completo.txt
/home/guilherme/log/completo.txt~
```

Ele usou o nome desses arquivos por causa da *globbing*. E se fizermos o `log-*` ? Tem alguém no diretório atual que possui esse nome?

```
> ls
arquivos.zip            falha~                mostra_idade~          temp
Desktop                 help                  Music                  Templates
Downloads               log completo.txt      Pictures               Videos
examples.desktop        loja                  sucesso
falha                   mostra_idade          sucesso~
```

Não tem!

Então, se fizermos `locate log-*` o que isso vai virar? Vamos testar?

Se dermos um "Enter" nisso ele não nos trará nada. Pois, `log-*` não significa nada. Não tem arquivo que possua esse padrão. Vamos verificar isso digitando `ls log-*` :

```
> ls log-*
ls: cannot access log-*: No such file or directory
```

Ficou a mesma coisa que se executássemos um `locate` com o argumento, só que esse argumento é vazio. Repare que se fosse sem argumento ele reclamava para nós, mas nós passamos um argumento, o problema é que ele é vazio e ele não nos traz nada, justamente, por ser vazio:

```
> locate
locate: no pattern to search for specified
```

Bastante cuidado quando passarmos para o `locate` algo que não faz sentido, por exemplo, `locate hahaha` ele não traz nada, ele não traz erro, entretanto, se executamos o `locate` "puro", ele traz uma mensagem de erro:

```
> locate
locate: no pattern to search for specified
```

Esse é o `locate` e esse é seu uso no dia a dia, extremamente útil, utilizamos ele muito para procurar arquivos que não lembramos exatamente onde eles ficam. Ou, mudamos de sistema operacional para outro e agora queremos buscar um arquivo de configuração que antes ficava em um diretório e agora fica em outro, usamos bastante o `locate` .

O `updatedb` é utilizado para atualizar o banco de dados e lembre-se que é necessário rodá-lo como `sudo` e que ele realiza uma busca em tudo. Ele possui dois modos de rodagens justamente pela questão de permissão que falaremos mais adiante.

O `locate` aceita asterisco, ponto de interrogação e colchetes e qual o cuidados que temos que ter com isso? Escapar! Seja com o uso da barra invertida, seja com aspas simples ou aspas duplas. E usando esses caracteres ele para de procurar em qualquer lugar do texto, do *path* de um arquivo, do caminho, mas ele passa a procurar no começo, então se você quiser procurar em qualquer local é necessário ter um asterisco no único da busca. Tudo isso é importante quando estamos utilizando o `locate` .

E com isso, vimos tudo o que necessitávamos do `man` , do `info` , `man pages` e `/usr/share/doc/` e do `locate` .