Automatização de scripts com o Sikuli

Script gráfico

Utilize imagens para montar scripts intuitivos automaticamente com o ambiente de scripts único do Sikuli.

por Dmitri Popov

Utilizando as capacidades de scripting do shell do Linux, é possível automatizar qualquer tarefa no sistema. Mesmo para quem não é um guru da programação, é possível escrever scripts que lidam com inúmeras tarefas corriqueiras – desde montar compartilhamentos remotos até fazer um backup.

Apesar de todo esse poder, produzir scripts shell possui uma séria limitação: os scripts são bons apenas para controlar e automatizar ferramentas de linha de comando e aplicativos gráficos que suportam argumentos pela linha de comando. Portanto, para automatizar ambientes desktop gráficos, como o Gnome e o KDE,

ou aplicativos baseados em interfaces gráficas, os scripts tradicionais não servem.

Eis que chega o *Sikuli* [1], um ambiente de scripting único que permite automatizar aplicativos gráficos com considerável facilidade. O Sikuli não depende de nenhuma API e utiliza como base simples telas com elementos gráficos. Basicamente, em vez de descrever onde clicar ou qual item do menu escolher, basta inserir no Sikuli uma tela da área específica diretamente no script.

O Sikuli analisa os padrões da imagem, encontra o elemento apropriado na interface gráfica e efetua a ação especificada. Nem é necessário dizer que isso simplifica o processo de produção de scripts. Na verdade, a abordagem do Sikuli é tão intuitiva que é possível começar a produzir scripts em questão de minutos, mesmo sem experiência em programação.

Primeiros passos

O Sikuli é escrito em Java. Então, a primeira coisa a ser feita é configurar o *Java Runtime Environment* no sistema. Além disso, será necessário instalar alguns outros pacotes. No Ubuntu, isso pode ser feito com o seguinte comando no terminal:

```
sudo apt-get install libcxxtools6
➥ libcxxtools-dev libhighgui1
➥ libhighgui-dev libcv1
```

Quando tudo estiver pronto, basta pegar a última versão do Sikuli e abrir o arquivo baixado no diretório de sua escolha (por exemplo, o diretório home). Em uma janela de terminal, vá para o diretório `Sikuli-IDE/`, e inicie o IDE Sikuli com o script `sikuli-ide.sh`. Outra maneira é iniciar o Sikuli com um duplo clique no arquivo `sikuli-ide.jar`, contanto que o arquivo `.jar` esteja associado ao Java Runtime.

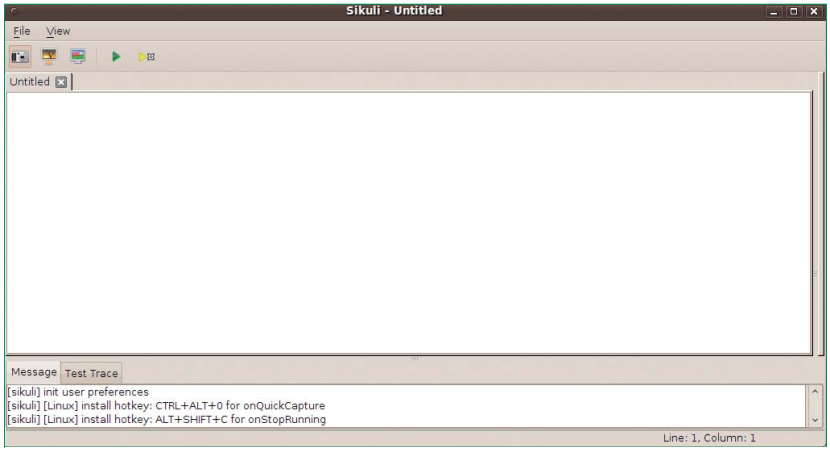

Figura 1 O IDE Sikuli possui apenas o essencial, com vários botões na barra de ferramentas.

A interface do Sikuli (**figura 1**) é fácil de entender. A barra de ferramentas principal oferece vários botões que dão acesso rápido a todas as funções essenciais do programa. Os três botões mais importantes são *Capture*, *Load* e *Run*. O botão *Capture* permite capturar uma tela do elemento desejado ou da área gráfica desejada; *Load* permite inserir uma tela já existente ao script atual. Como se pode supor, o botão *Run* executa o script aberto. Graças ao suporte a abas, é possível usar o IDE Sikuli para abrir e gerenciar vários scripts ao mesmo tempo.

Ao trabalhar com o Sikuli, é bom ter algumas coisas em mente. Primeiro, o Sikuli não se dá bem com múltiplos monitores. Então, para ter certeza de que o script funcionará como esperado, desconecte todos os monitores externos da máquina (exceto o principal, é claro).

No início, a maioria dos meus scripts se recusou a funcionar corretamente. O motivo é que eu havia habilitado a opção do botão esquerdo do mouse, e todas as ações eram interpretadas como cliques com o botão direito. Então, para os usuários canhotos, a correção é simples. Use a ação de clique com o botão direito em vez de clique, e vice-versa.

A versão do Sikuli para Linux ainda não suporta atalhos de teclado. Portanto, é necessário usar o botão *Capture* para capturar as telas. Como alternativa, é possível usar um utilitário externo como o *Shutter* para capturar as telas e depois importá-las para seu script usando o botão *Load*. O uso de uma ferramenta dedicada à captura de telas oferece maior controle e acelera todo o processo. No entanto, lembre-se de que todas as telas precisam estar no formato PNG.

Para começar, criei um script simples que clica no ícone do mini-aplicativo *Deskbar* no painel e digita "Hello world!" no campo de busca. Obviamente, para esse script funcionar, é preciso antes instalar os pacotes do mini-aplicativo Deskbar e adicioná-lo ao painel. Um script do Sikuli consiste em uma série de ações tais como *click*, *wait*, *type* e assim por diante. Cada ação pode usar uma tela que espe-

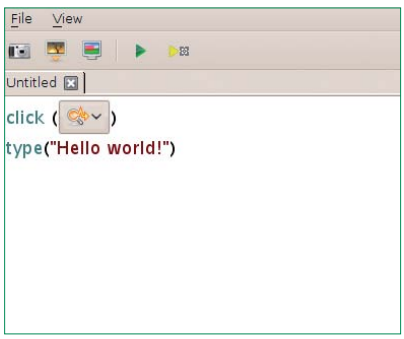

Figura 2 Script *Hello world!*.

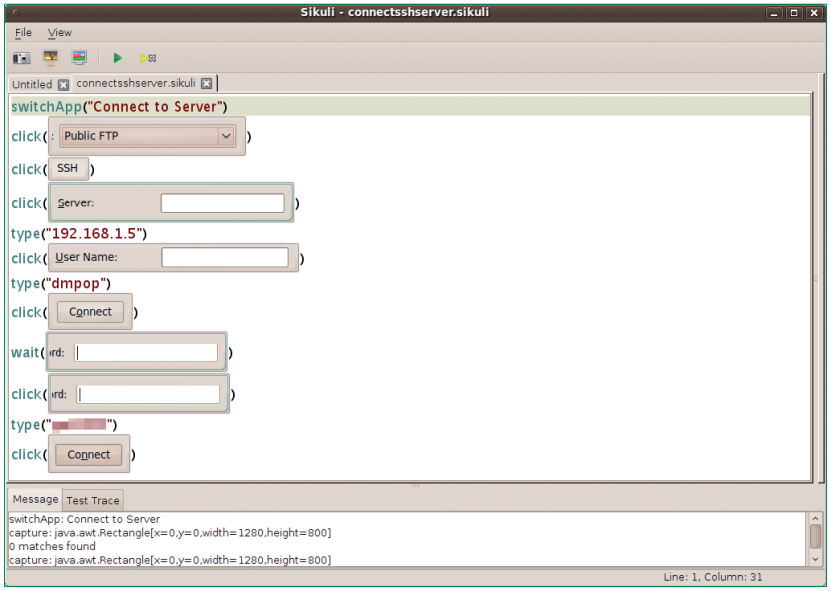

Figura 3 Este script automatiza o processo de montar um compartilhamento remoto SSH.

cifica o elemento ou a área gráfica alvo. Por exemplo, se o script tiver que clicar em um botão específico de um dado aplicativo, é preciso adicionar a ação *click* seguida pela tela do botão. Nesse caso, o script deverá fazer duas coisas: clicar no mini-aplicativo Deskbar no painel e digitar o texto "Hello world!" no campo de busca.

Para conseguir isso, é necessário especificar duas etapas: *click*, com a tela do botão do Deskbar, e *type*, com o texto *Hello world!*, como mostra a **figura 2**. Em seguida, clique no botão *Run* na barra de ferramentas principal do Sikuli e admire a magia do seu script em ação.

Agora que já mostrei como o Sikuli funciona, criarei um script que finalmente faz algo útil, como montar um compartilhamento remoto via SSH usando a ferramenta *Locais | Conectar ao servidor*. O script completo está na **figura 3** e a maioria dos passos mostrados é óbvia.

Uma série de ações *click and type* são utilizadas para fornecer os campos na caixa de diálogo e digitar os textos fornecidos, tais como endereço do servidor e nome do usuário. No entanto, dois passos pedem uma análise mais atenta. Como o nome sugere, a ação *switchApp* direciona o script para o aplicativo desejado (neste caso, o utilitário *Conectar ao servidor*). Para selecionar um item na lista *drop-down*, são necessárias duas ações *click*: a primeira clica na lista drop-down propriamente dita (isto é, a lista *Tipo de servidor*), enquanto que o segundo clica no item da lista (isto é, *SSH*). Para determinar o item da lista usando o Sikuli, é preciso ajustar o tempo de captura: assim, haverá tempo suficiente para clicar na lista antes que o Sikuli entre no modo de captura de tela. Para ajustar esse tempo de captura, acesse o menu *File | Preferences* e especifique o tempo, em segundos, no campo *Capture delay*.

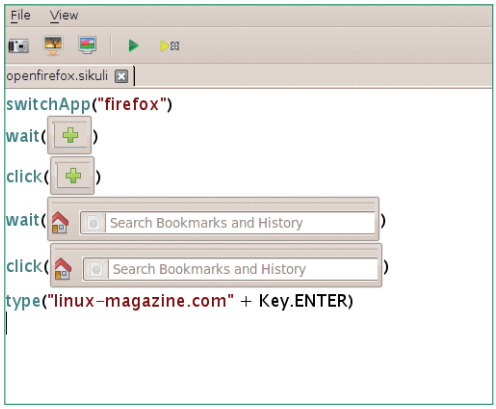

Figura 4 Em vez das ações *click*...

Além da capacidade de digitar strings específicas, o Sikuli também trabalha com o teclado e com teclas modificadas, que é uma maneira mais eficiente de automatizar aplicativos. Por exemplo, o script da **figura 4**. Esse script entra no navegador Firefox, clica no botão *Nova Aba*, insere a URL dada e tecla **[Enter]**. Para o último passo, o script usa o argumento `Key.ENTER`.

Por suportar modificadores de teclado, é possível substituir ações como clicar no botão *Nova Aba* pelo comando `type("t", KEY_CTRL)`, que emula o atalho **[Ctrl]+[T]** do teclado (como mostra a **figura 5**).

Outros modificadores de teclado suportados incluem `KEY_ALT` (a tecla **[Alt]**), `KEY_META` (a tecla **[Meta]** ou Windows) e `KEY_SHIFT` (a tecla **[Shift]**).

O uso da ação *click* possibilita também selecionar ou tirar a seleção de uma caixa em uma janela de diálogo. Porém, e se a janela contiver várias caixas e for necessário selecionar todas de uma vez? É nesse ponto que a ação *findAll* é útil. Essa ação encontra todas as ocorrências de uma imagem especificada. Então, se uma tela com uma caixa for usada juntamente com a ação *findAll*, todas as caixas serão encontradas na janela especificada. Agora, essa ação precisa entrar em um *loop* `for...in` para fazer o script passar por todas as caixas encontradas e selecioná-las.

O script da **figura 6** seleciona todas as caixas na caixa de diálogo *Preferências do Gerenciamento de Energia*. Caso você já tenha traba-

Figura 5 ...use a ação *type* com modificadores de teclado para emular atalhos de teclado.

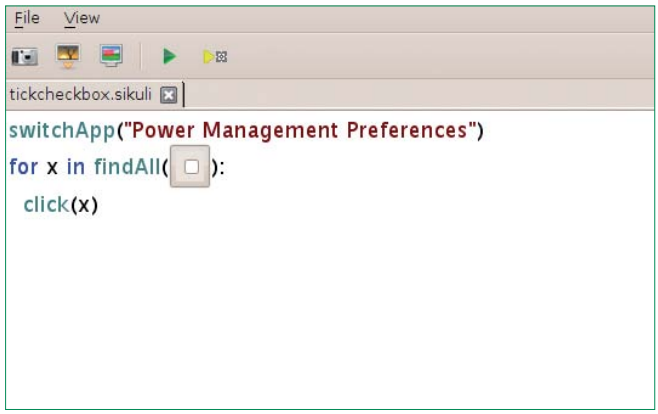

Figura 6 Selecionando todas as caixas.

lhado coma linguagem Python, o código do script será familiar. Isso não é uma coincidência, pois o Sikuli usa Jython como base. Não há necessidade de habilidades de programação para trabalhar com o Sikuli, mas certo conhecimento de Python pode ajudar no uso mais avançado do programa.

Normalmente, o Sikuli faz um bom trabalho de reconhecimento de padrões de imagens nas telas, mas seu IDE oferece um recurso útil que permite ajustar e testar a precisão do reconhecimento dos elementos da tela. Para isso, clique na tela desejada no script, para que o Sikuli abra uma janela de visualização, onde as áreas que correspondem ao padrão da imagem na tela estão marcadas com retângulos vermelhos. Isso pode ajudar na solução de problemas do script, caso as áreas corretas não estejam sendo atingidas na interface especificada. Além disso, é possível usar o *Slider Similarity* para precisar o reconhecimento. Quando o script estiver pronto, ele deverá ser exportado como um pacote executável `.skl` selecionando o comando *Export executable* no menu *File*. Então, ele poderá ser executado sem ser aberto no IDE Sikuli usando o seguinte comando no terminal:

```
caminho/do/sikuli/sikuli-ide.sh
➥ script.skl
```

Substitua `/caminho/do/sikuli` pelo verdadeiro caminho até o diretório do IDE Sikuli, e `script.skl` com o nome do script que será executado.

Trocando em miúdos

O Sikuli é um projeto fascinante e com enorme potencial. O projeto ainda está engatinhando e por isso há espaço para melhorias. Neste aspecto, a documentação dos recursos do Sikuli ainda é deficiente. No momento, o melhor método para conhecer o Sikuli é usá-lo e descobrir suas possibilidades escrevendo scripts. Para começar, é possível usar a seção "*Sikuli Script Commands for Jython*" [2] na documentação e no blog do projeto [3]. ■

Mais informações

[1] Sikuli: http://groups.csail.mit.edu/uid/sikuli/

[2] Comandos do Sikuli para Jython: http://sikuli.org/documentation.shtml#doc/pythondoc-python.edu.mit.csail.uid.Sikuli.html

[3] Blog oficial do Sikuli: http://blog.sikuli.org/

Sobre o autor

Dmitri Popov é formado em letras (idioma russo) e lingüística computacional; há vários anos trabalha como tradutor técnico e colaborador free-lancer. Já publicou mais de 500 artigos sobre software de produtividade, computação móvel, aplicativos web e outros tópicos relacionados à informática. Seus artigos já apareceram em sites e revistas da Dinamarca, Inglaterra, EUA, Alemanha, Rússia e, agora, do Brasil.

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/3467