Universidade Federal do Rio Grande do Norte

Tecnologia da Informação / Ciências da Computação

Relatório Final

Sistemas Operacionais

Clipboard Manager

Professores:

Edgar de Faria Corrêa

Ivanovich Medeiros Dantas da Silva

Alunos:

João Santana Belau

Patrícia Pontes Cruz

Natal

Rio Grande do Norte

10/06/2016

Resumo

Este é o relatório referente ao projeto final da disciplina de Sistemas Operacionais DIM0615 ministrada pelos professores Edgard Corrêa e Ivavonitch Medeiros.

Temos como objetivo estabelecer conexões com os conteúdos ministrados durante o período letivo e criar um software inovador e/ou melhorar algo já existente que faça a utilização das propriedades do sistema operacional do usuário.

Assumimos então a responsabilidade da criação de um Software gerenciador de Clipboard, como sugerido pelo professor Ivanovitch. Para produzir tal trabalho, fizemos a utilização da IDE QtCreator v5.5.1 no sistema operacional Linux Ubuntu 14.04.1 LTS. Cada sistema operacional possui uma maneira diferente de lidar com o clipboard e, nas próximas páginas, esse processo será explicado passo a passo.

Introdução

O Sistema Operacional de um computador é o principal responsável pela iteração do usuário com o dispositivo. É ele quem traz programas, interface gráfica (ou textual) e possibilita uma comunicação de mais alto nível com o disposítivo. Drivers, gerenciamento de dispositivos de entrada e saída, comunicação correta entre devices externos, alocação e paginação de memória e gerenciamento de processos são bons exemplos das responsabilidades de um SO. Dentre as facilidades que um bom sistema operacional gera para o seu usuário, o clipboard é uma delas.

O clipboard é o responsável pela área de transferência, ou seja, é ele quem “armazena” textos, imagens e outros tipos de arquivos quando pressionamos o famoso “Ctrl+C” ao selecionar algo. Um dos problemas do clipboard é o fato dele ser temporário e não haver, em nenhum lugar do SO, um local em que se possa encontrar os objetos que estavam no clipboard antes dele ser alterado.

O nosso trabalho se encarrega de criar um gerenciador de clipboard que possa ficar em segundo plano e capturar, através de um sinal, os eventos de inserção no clipboard, salvar os dados copiados em estruturas referentes aos seus tipos (texto e imagem) e exibir ao usuário um histórico de 10 textos copiados e de 5 imagens. Nosso aplicativo também mostra o tamanho da imagem em pixels e em bytes. E, claro, não existe a necessidade de um gerenciador que não permita mandar novamente os dados para o clipboard, tornando possível colocá-lo novamente na área de transferência.

Desenvolvimento

Para o desenvolvimento do trabalho, foi necessário, em um primeiro momento, um estudo de como funciona o clipboard dos sistemas operacionais mais recorrentes (Ubuntu e Microsoft Windows). Viu-se que os sistemas operacionais não possuem em si a função do clipboard, mas que essa ferramenta advém do provedor gráfico ou de algum outro software dedicado que roda em uma thread em segundo plano.

Um sistema operacional com interface gráfica possui um provedor (ou protocolo) responsável por gerar os gráficos com os quais o sistema irá trabalhar. Nos sistemas Windows, o clipboard é feito através de um software proprietário que foi modificado nas versões mais recentes do sistema. Antigamente era possível abrir o programa “clipbrd.exe” e ter acesso a um gerenciador de clipboard do sistema. Já no Ubuntu (alvo desse trabalho, ainda que no final tenhamos contemplado os dois sistemas), o serviço é feito pelo provedor gráfico “X.Org”. O “X.Org” (também conhecido como “X11”) é comumente utilizado em sistemas operacionais *Unix* based e é responsável por toda a representação gráfica. Ele é um dos fatores de compatibilidade entre os diferentes gerenciadores de janelas espalhados pelos diferentes sistemas baseados em Unix (Gnome, KDE, Lightdm, etc).

A direita temos uma imagem de uma versão antiga do X.Org, provendo os gráficos para as janelas do Gnome em um ambiente gráfico, mas ainda prioritariamente textual.

Figura 1. Antigo X.Org

O Gerenciador

Desenvolvemos o gerenciador de clipboard utilizando o software QtCreator. Esta IDE possui bibliotecas prontas para o clipboard que implementa a integração com o clipboard do sistema operacional em que está rodando, ela se chama QClipboard.

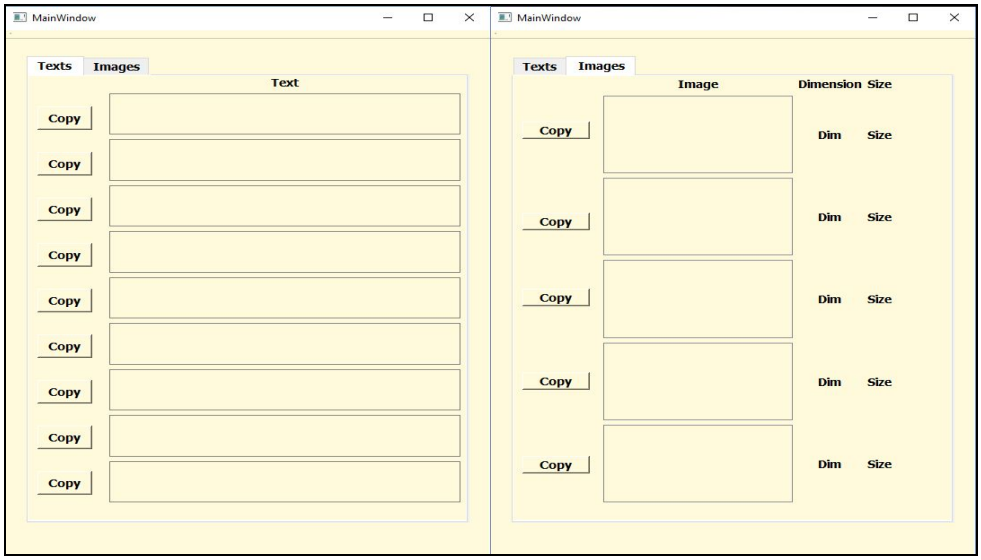

Figura 2. Interface Gráfica

Para evitar que o programa seja fechado erroneamente ou que incomode o usuário a sempre tê-lo aberto, utilizamos a biblioteca QSystemTrayIcon, também do QtCreator, para que o programa continue a rodar em background, possibilitando que o usuário abra e feche a interface gráfica quando lhe for conveniente. Segue abaixo uma imagem mostrando o icone, um cogumelo com uma lupa, utilizado no trabalho para demarcar o uso do programa em background e também para iteragir com o usuário quando necessário.

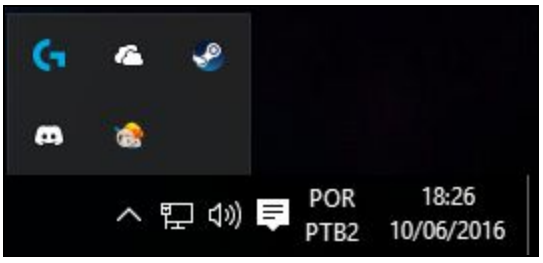

Figura 3. Ícone do programa (cogumelo com a lupa)

Implementação

A criação do código se deu de forma flúida e eficiente. Utilizando as bibliotecas prontas como base e alguns tutoriais, o grupo não apresentou grandes dificuldades na em sua implementação. Alguns pontos importantes que as bibliotecas já tratam é a cópia de dados protegidos (ex: senhas de banco), qual sistema operacional o usuário está usando para utilizar o método de captura de clipboard específico do mesmo e o funcionamento do programa em background quando a interface é fechada.

Implementamos o Clipboard utilizando a aplicação QApplication::Clipboard(). Essa aplicação parte de uma implementação do Qt que permite a declaração de um ponteiro para o clipboard, apontando para o dado armazenado naquele instante. Fizemos uma função que funciona com o sinal de mudança no clipboard, isto é, quando o usuário aperta “Ctrl+C” ou copia com o mouse select, o clipboard muda seu estado e dá *trigger* na chamada da função que trata o dado e o armazena. Esse dado vem do clipboard em formato “MimeData”, esse tipo de dado é utilizado com multiproposito, pois a área de transferência serve para diversos tipos de dados, desde textos à imagens e arquivos (ainda que muitas vezes arquivos sejam tratados como strings que representam o caminho até ele). Neste trabalho apenas utilizaremos textos e imagens.

A interface gráfica foi pensada para ser simples e de acordo com uma análise prévia do comportamento de um usuário normal ao utilizar o “ctrl+c” e “ctrl+v”. Observamos a necessidade de uso dos dados copiados anteriormente. Percebemos que armazenar 9 arquivos texto e 5 imagens copiadas são o suficiente para cobrir as necessidades do usuário, então planejamos a interface com duas abas independentes, uma para texto e outra para imagens, com a quantidade de espaços equivalente à estudada, como observados na figura 2. Na parte de imagens, foram adicionados os campos de tamanho e proporções apenas para acrescentar informações extras ao usuário. Outro comportamento observado foi, ao copiar um texto, o usuário acaba pressionando repetidamente em um curto intervalo de tempo o “ctrl+c”, de forma a querer ter certeza que a cópia foi executada com sucesso. Para tratar esse caso,

enquanto o dado copiado previamente for igual ao copiado no momento, este é ignorado e apenas o prévio é mostrado na interface.

Tinhamos em mente deixar o usuário escolher os dados que ele tem interesse de obter novamente ao clicar duas vezes em cima dele. Porém, como precisaríamos criar métodos de verificação de clique nos tipos de objetos que utilizamos para informar os textos e imagens, tornou-se complicada a sua implementação. Tinhamos como objetivo a funcionalidade total do uso do clipboard, pois ela era de maior importância do que facilidades da interface, por isso demos preferência ao uso de botões simples.

No ambito da implementação do tray icon, implementamos a execução do programa em background, a criação do ícone com a imagem préviamente escolhida, como observado na figura 3, e duas opções, Restore e Exit, que são mostradas ao usuário quando ele selecionar o icone. O restore abre a interface gráfica quando esta estiver fechada e o Exit finaliza o programa. É interessante notar que a aplicação (que por si só é uma thread) continua rodando mesmo com a interface gráfica fechada, tornando desnecessária a implementação de uma thread específica para cobrir este caso.

Conclusão

O projeto abre margem para implementações bastante interessantes. Em futuras melhorias ao projeto (que devem acontecer nos próximos meses), pensamos na implementação de um servidor de dados Mime que envia os dados do clipboard para um celular cliente.

Trabalhamos, durante a disciplina, com diversas sub-rotinas do sistema. Threads, sinais, sockets, memória, processos e drivers foram estudados pela turma durante o período letívo. Neste trabalho aplicamos os conceitos de threads, sinais e o conhecimento de parte mais profunda do sistema. Utilizamos também a IDE QtCreator que foi bastante incentivada durante o curso.

Acreditamos que a execução deste projeto nos ajudou a nos aprofundar nos conceitos utilizados enquanto estudávamos a maneira como a área de transferência funciona em seu núcleo bem como a integração de um software ao sistema operacional em si, adquirindo assim novos conhecimentos e capacidades. Também foi interessante para terminarmos de nos familiarizar com a IDE utilizada.

O estado final do projeto foi considerado bastante satisfatório. Todas as ideias e requisitos foram preenchidos de forma que o programa funciona perfeitamente como sua ideia foi concebida.

Codigos extras

Os códigos extras e auxiliares, inclusive este relatório, podem ser observados abaixo no nosso GitHub: https://github.com/Pekorishia/Clipboard-Manager.

Referências

https://en.wikipedia.org/wiki/X_Window_selection

http://doc.qt.io/qt-5/qsystemtrayicon.html

http://doc.qt.io/qt-4.8/qclipboard.html

https://www.x.org/wiki/