onto de Controle 04

Sistemas Embarcados

Leonardo Brandão Borges de Freitas
RA: 14/0025197
Universidade de Brasília, Campus Gama
Brasília, Brasil
leonardobbfga@gmail.com

Ygor Pereira Borgonove
RA: 14/0166408
Universidade de Brasília, Campus Gama
Brasília Brasil
ygor.borgonove@hotmail.com

Resumo – ***Desenvolvimento de um pequeno equipamento constituído por um display para interfaceamento gráfico, um HD para armazenamento de arquivos e um Raspberry Pi para embarcar procedimentos de upload e downloads de dados para uma nuvem física, offline e com alcance local de roteiamento, em que o usuário so terá acesso com autenticação de login.***

Palavras-chave — Skullpi, Nuvem, Raspberry Pi, Download, Upload, Segurança, Sistema Embarcado.

I. Introdução

Armazenamento e transferência de dados são palavras chaves em nosso cotidiano. Com o avanço da tecnologia digital, geramos cada vez mais material a ser compartilhado e distribuído. Para tanto, expandir o a capacidade de servidores e acelerar o compartilhamento online de arquivos é foco de muitos desenvolvedores como a Google, Youtube, Amazon e vários outros.

Figura 1: Logo Google Drive

Em meio a tanta demanda criou-se o serviço de nuvem, em que são fornecidos servidores para armazenamentos de arquivos em banco de dados, backup, dentre outros serviços. Atualmente os maiores serviços de núvem prestados são: Google Drive, Dropbox, One Drive e iCloud. Existem, basicamente, três tipo de nuvens: IaaS, PaaS, SaaS.

IaaS é a mais básica das três em que o usuário aluga a infraestrutura de TI, servidores e máquinas virtuais, são as mais fornecidas pelas empresas citadas anteriormente.

PaaS, plataforma como serviço, é usada para serviços de computação em nuvem que fornecem um ambiente para desenvolvimento, testes, gerenciamento, dentre outras aplicações para empresas. Foi criada para auxiliar os desenvolvedores a criar aplicativos.

SaaS, software como serviço, basicamente é uma forma de distribuição e comercialização de software. Ao utilizar o SaaS as empresas contratantes não precisam se preocupar com instalação, manutenção e atualização de hardware ou software, só precisaria de uma conexão com a internet para gerenciar o que estivesse na núvem. Muito utilizado para serviço de assinaturas.

Em meio globalizado a segurança de arquivos está cada vez mais delicada e com alto custo. Muitas vezes nos deparamos com situações de vazamentos de arquivos pessoais em que o usuário paga caro por um produto ou serviço para armazenamento de seus materiais e devido a falhas na segurança, alguns mal intencionados conseguem ter acesso aos seus arquivos pessoais podendo causar danos irreparáveis ao usuário. Um grande exemplo dessa quebra de segurança foi o que aconteceu com usuários da Apple, pessoas famosas tiveram fotos particulares expostas na internet recentemente. Uma simples solução para esse

problema seria uma nuvem offline com caráter físico que só poderia ser acessada em curto alcance e por poucos usuários cadastrados, evitando assim problemas de quebra de segurança e vazamento de dados.

II. Desenvolvimento

Para tanto, será construído um sistema de nuvem com um software chamado Skullpi codificado em linguagem C, desenvolvido no sistema operacional Linux Raspbian, embarcado em um Raspberry Pi 3 Model B, interfaceado por um display de 7" touchscreen, com armazenamento em um cartão micro SD de 16 GB, com adaptação para HD externo caso mais memoria seja demandada.

O funcionamento do Skullpi tem 4 etapas associadas (A, B, C e D):

A. Conexão:

O Skullpi é um aparelho eletrônico pessoal, contudo, sua utilidade pode ser compartilhada de forma segura. Devido a isso o proprietário deve definir uma senha para a conexão wireless, a qual ficará visível a todo momento, no entanto só terão acesso à nuvem quem souber a senha do wi-fi.

Com a conexão estabelecida o usuário pode executar o programa SkullMain, o qual o apresentará o menu principal do user interface.

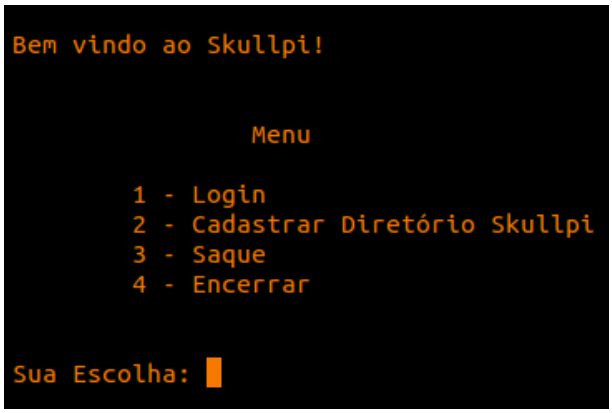

Figura 2 – Menu User Interface (SkullMain)

B. Login:

O segundo passo é a etapa de login, que possui duas instâncias: cadastro e autenticação.

O procedimento de cadastro pede ao usuário o endereço da sua pasta pessoal (/home/nome_de_usuário) e uma senha a ser criada, os quais serão seu passaporte de acesso para o Skullpi. Essa etapa também é responsável por criar dois diretórios: um na máquina do usuário (/home/nome_de_usuário/Skullpi), a qual será utilizada para o download e upload de arquivos para a nuvem, e outra dentro do hd do raspberry (/home/pi/Skullpi/nome_de_usuário) a qual armazenará os arquivos enviados. Desta forma, o processo de cadastro linka o login com apenas um diretório para gerenciamento de dados, conferindo maior segurança e organização.

A autenticação é a fase necessária para acionar o menu de Gerenciamento do programa. Ela pedirá o nome de usuário e a senha e se for verificado a existência desses dados na sessão de cadastrados no raspberry o usuário passa para a etapa de gerenciamento de arquivos em seus diretórios.

C. Gerenciamento:

Aqui o usuário comum tem permissões de gerenciamento apenas em seu diretório, sendo que o superusuário pode gerenciar todos os diretórios e arquivos presentes em seu Skullpi.

De forma simples e intuitiva o menu possui 3 opções: Upload, Download e Encerrar.

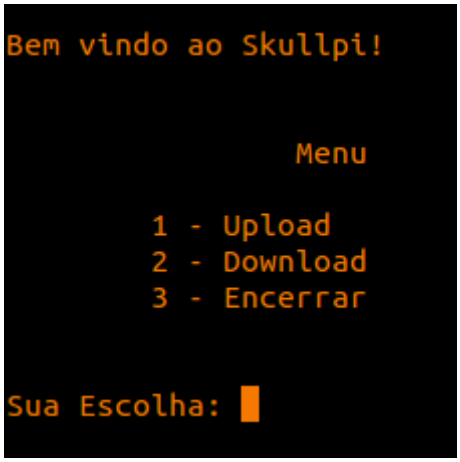

Figura 3 – menu de Gerenciamento (login efetivado)

Para o upload de dados, deve-se colocar os arquivos, que deseja armazenar na nuvem, dentro da pasta Skullpi

(/home/nome_de_usuário/Skullpi) e selecionar a primeira opção do menu. Se o procedimento ocorrer de forma correta, os dados são armazenados no HD do raspberry e a pasta Skullpi passa por um processo de "refresh" o qual apaga os arquivos que já foram enviados, deixando a pasta vazia.

Ao selecionar a segunda opção do menu o usuário faz o download de todos os seus arquivos contidos no seu diretório do Skullpi do raspberry. Esses arquivos aparecem na pasta Skullpi na máquina do usuário.

A última opção encerra o login e retorna o programa para o menu principal e além disso, aqui o usuário também define se seu diretório ficará em modo de Saque ou não.

D. Saque:

Ao selecionar o Saque no menu principal o programa pede a pasta pessoal do usuário (/home/nome_de_usuário) e também qual o diretório que será "sacado", ou seja, qualquer pessoa que tenha acesso ao wi-fi do raspberry, pode utilizar dessa opção. No entanto, o diretório cadastrado só entra nesse modo se seu proprietário autorizar. Para tanto, o Skullpi só mostrará na interface gráfica as pastas que tem o modo saque ativado.

Um PIN diferente é gerado e mostrado no display toda vez que essa opção é escolhida e o download dos arquivos só é efetivado inserindo o PIN corretamente, se isso ocorrer, o diretório escolhido para download surgirá em sua pasta pessoal.

Portanto, o modo saque tem a mesma função do Download, no entanto, o login não é necessário e apenas diretórios autorizadas podem ser baixados nesse modo.

E. Comunicação e transferência de dados

A comunicação entre a máquina do usuário e o Raspberry embarcado com o software Skullpi é realizada pela leitura e escrita de arquivos através das funções **open()**, **read()**, **write()** e **close()** presentes nas bibliotecas **fcntl.h** e **unistd.h** que seguem as normas POSIX (IEEE 1003). Também utiliza-se a função **system()** da biblioteca **stdlib.h** para execução de algumas subrotinas em linguagem Unix.

A transferência de dados se dá através do protocolo de rede criptografado **Secure Shell** ou **SSH,** o qual utiliza modelo Cliente-Servidor para comunicação e algumas funções como **SCP** (Secure Copy) para a transferência de dados de uma máquina para a outra.

III. Descrição de Software

O funcionamento do Skullpi é dividido em dois programas, um que fica compilado e rodando no Raspberry e outro que fica como executável na máquina do usuário.

A. SkullBian v2.0:

Esse é o programa que está em constante execução dentro do Raspberry. Na versão 2.0, esse software executa 13 processos em paralelo, sendo 12 processos filhos gerados por um processo pai. Os quais se comunicam através de sinais (SIGUSR1) transmitidos pelo processo pai designando funções aos filhos.

O processo pai está sempre acordado e checando envios de arquivos do processo de **upload** do **SkullMain.** Os filhos, por sua vez, estão em estado de sono esperando o sinal do pai. Os 10 primeiros filhos (pid[0] a pid[9]) são exclusivos para a manutenção dos 10 diretórios presentes no **Skullpi**, realizando compactação, descompactação e transporte de arquivos, o décimo primeiro filho (pid[10]) realiza o cadastro de novos usuários e o décimo segundo filho autentica quais pastas estão disponíveis para saque.

Além da rotina de processos, esse código conta com 3 tipos de funções:

- Funções Alfa: Inicialização e preparação de arquivos e diretórios necessários para o funcionamento geral do programa. Inclui funções que verificam a existência do diretório e banco de dados Skullpi, bem como o arquivo que armazenam os números dos diretórios relacionando-os com os usuários cadastrados e também o arquivo que lista quais pastas podem ser sacadas por usuários não cadastrados (cadastrados.txt e emsaque.txt).

- Funções Beta: Gerenciamento e coordenação de processos. Inclui as funções de cada processo filho, bem como função de cadastro e de autenticação de saque.

- Funções Gama: Tratamento e transporte de arquivos. Inclui funções para compactar e descompactar arquivos alocando-os em seu devido local.

Obs.: O programa em C, junto com os demais scripts necessários para a execução do programa estarão disponíveis no git e o link estará nas referências.

Obs2.: O código está todo comentado e bem explicado em seu decorrer. Para tanto, recomenda-se fortemente que dê uma lida no arquivo disponibilizado.

B. SkullpiMain v2.0:

O Programa principal de interface com o usuário apresenta as opções de menu para que esse possa se cadastrar, ou logar no gerenciamento de seu diretório no sistema Skullpi.

O menu é uma função do tipo inteiro que retorna para a função principal (main) o resultado da escolha do usuário. Sua visualização e manipulação depende diretamente de uma variável inteira global chamada **flag_login**, a qual indica se o usuário está logado ou não com seu diretório Skullpi. Desta forma, quando iniciado o programa o menu se mostra de acordo coma figura 2.0 e quando logado o usuário terá acesso ao menu de gerenciamento (figura 2.1).

A subrotina **cadastrar()** é uma função do tipo void que quando executada realiza o cadastramento de um novo usuário. Este deve inserir qual o nome de sua pasta principal (/home/nome_de_usuário) e a criação de uma senha, possibilitando o funcionamento das demais subrotinas do programa **SkullMain**. Além disso, através do protocolo **ssh** essa função envia um arquivo txt (cad.txt) contendo o usuário e a senha do novo cadastro realizado para o aparato **Skullpi** que o armazena no arquivo **cadastrados.txt** no software que está rodando no Raspberry, para futuramente ser requisitado em procedimento de login.

A função **login()** é do tipo void e realiza um procedimento de confirmação, o qual permite acesso ao menu de gerenciamento do diretório dentro do **Skullpi**. Através da função **strcmp()** o programa compara a entrada inserida pelo usuário (usuário e senha) com o arquivo **cadastrados.txt,** que é "puxado", através do protocolo **ssh,** do **Skullpi**, verificando linha por linha ate o final do arquivo. A **flag_login** se torna 1 se a comparação for verdadeira, ou seja, se o nome de usuário e senha estiverem cadastrados e corretos.

A função **upload()** do tipo void executa uma sequência de chamadas **system()** para executar comandos diretamente no terminal pela linguagem **Unix**. Primeiramente ela verifica qual o número de diretório relacionado ao cadastro (**usr**), para que quando enviada para o diretório **Skullpi** do raspberry essa seja armazenada de forma correta. Logo em seguida a pasta é compactada e enviada via **ssh** para a nuvem. Depois a **upload()** exclui o compactado e faz o procedimento de "refresh" na pasta **Skullpi** da máquina do usuário, limpando-a.

A função **download()** do tipo void executa uma sequência de chamadas **system()** para executar comandos diretamente no terminal pela linguam **Unix.** Através do protocolo **ssh** essa função requisita do banco de dados do raspberry os compactados de arquivos (pak.tar) os quais contem os dados que estão armazenados no diretório **Skullpi.** Da mesma maneira que a função upload, o download diferencia qual compactado deve baixar pelo número do diretório cadastrado (**usr**) proveniente do arquivo cadastrados.txt.

A função **main()** do software executa apenas uma loop infinito (while) com estrutura de decisão (Switch/case) para realizar a chamada das funções em relação à escolha do usuário na função **menu()**. Além disso, através da diretiva **signal()** ela atribui o sinal de cancelamento (Ctrl + C) à subrotina **encerrar()** que é uma função do tipo void que apenas trata o sinal fechando os descritores de arquivos que poderiam estar abertos.

Para o funcionamento de todas as funções utilizadas pelo software algumas

bibliotecas e variáveis globais para manipulação de arquivos estão presentes no header do código.

Obs.: todo o código será disponibilizado no git junto ao relatório e o link estará nas referências.

IV. Descrição de Hardware

O hardware que será utilizado será o Raspberry Pi 3 e a uma tela touchscreen de 7'' desenvolvida pela empresa que fabrica o Raspberry. Essa tela tem uma resolução de 800x600 e acoplada a ela existe um shield que facilita a comunicação entre o Raspberry e a tela. Para ligá-la bastam dois jumpers, VCC e GND comum entre o Rasperry e a tela e um cabo DSI que faz a comunicação entre a tela e o Raspberry. A tela e o Raspberry puxam juntos, em média, de 1 a 1,5 Ampères. Será utilizado também um computador para fazer as transferências de arquivos.

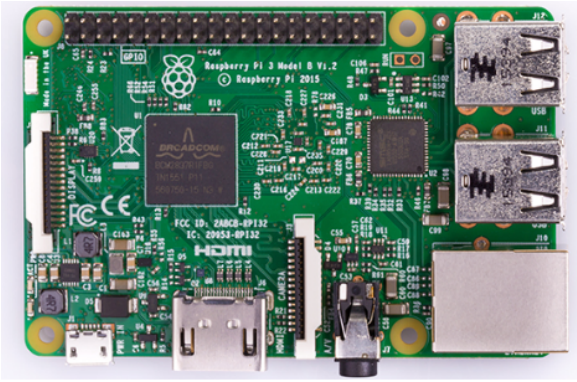

Figura 4: Raspberry Pi 3 Model B

Figura 5: Tela touchscreen 7"

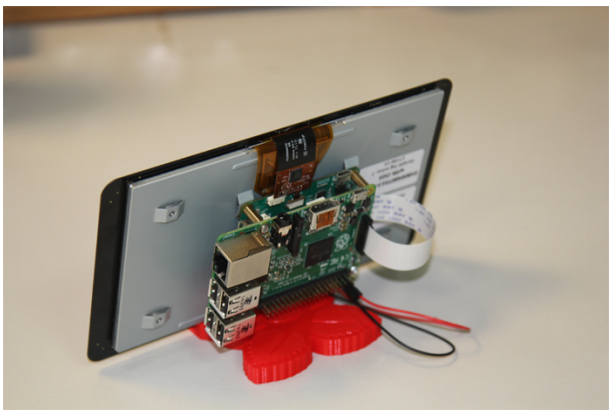

Figura 6: Tela e Raspberry Pi 3 acopladas

V. Resultados

Essa versão 2.0 do programa Skullpi já supri 95% da ideia geral do que será esse projeto. As funções de gerenciamento de usuários e de transferência de arquivos de (cadastro, login, upload, download) já estão disponíveis e 100% operantes, bem como toda logística de processamento interno ao raspberry. No entanto, nesse ponto o Skullpi assiste a apenas 10 usuários cadastrados devido ao número finitos de processos que o programa tem. Sinais foram adaptados ao software, melhorando seu desempenho e resposta aos pedidos dos usuários. Contudo, para finalização total do projeto precisamos melhora o procedimento de Saque que ainda não está totalmente operante. Outro ponto a ser melhorado é a forma que o processo pai enxerga as solicitações dos usuários: no momento ele utiliza o retorno da função POSIX **open()** para saber qual arquivo chegou e assim sinalizar aos filhos. O próximo passo é analisar o diretório /tmp (o qual armazena os pedidos e arquivos enviados pelos usuários) de uma maneira mais eficiente ao envés da utilização da função **open()**. A ideia ainda está em desenvolvimento.

VI. Conclusão

Considerando que o projeto está quase pronto, faltando somente algumas otimizações

de código, a função saque e interface gráfica que será feita utilizando a biblioteca Dialog implementada diretamente no terminal, Com isso podemos concluir que 4/5 do projeto está pronto.

Referências

[1]https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/

[2] https://www.salesforce.com/br/saas/

[3]https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

- SkullpiBian v2.0: https://github.com/leobbf/Sistemas_Embarcados/blob/master/SkullBianv2.c

- SkullpiMain v2.0: https://github.com/leobbf/Sistemas_Embarcados/blob/master/SkullMainv2.c