UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

Emanoel Dantas Pereira
Matheus Rangel de Melo

RELATÓRIO DO TRABALHO 1.2 DE PROJETO DE SISTEMAS OPERACIONAIS (DIM0615)

Natal-RN, 5 de setembro de 2018

1. INTRODUÇÃO

O trabalho 1.2 da disciplina de Projeto de Sistemas Operacionais (código DIM0615), ministrada pelo professor doutor Edgard de faria Corrêa no primeiro semestre do ano 2018, tratava-se, de acordo com a especificação disponibilizada na turma virtual da disciplina, da implementação de dois programas, um que monitorasse e outro que consumisse um determinado recurso (CPU, memória, etc.) da Beagle Bone Board. Na implementação do programa de monitoramento, as GPIOs deveriam ser usadas para sinalizar, através de LEDs, o percentual de uso desse recurso da seguinte forma: um LED verde acenderia se o uso fosse de até 25%, um amarelo, se o consumo estivesse entre 25% e 50%, um vermelho, para o caso de estar sendo consumido entre 50% e 75% do recurso, e para um uso acima de 75% todos deveriam ficar piscando. Neste último caso, um botão deveria ser pressionado para matar o processo que estava consumindo maior parte do recurso. Quando isto acontecesse, todos os LEDs deveriam ficar apagados por x segundos e então o programa iniciaria o monitoramento novamente. Assim, vejamos adiante os detalhes da implementação deste trabalho.

2. DESENVOLVIMENTO

Na implementação foi usada a linguagem C++ e todos os códigos encontram-se em https://github.com/Matheus-Rangel/PSO/tree/master/Projeto2.

2.1. SOLUÇÃO ENCONTRADA

Inicialmente decidimos que o recurso que seria monitorado era a memória. Após isso, criamos o programa MemoryLeaker/memoryLeaker.cpp que consumia de forma crescente a memória RAM da máquina. Sua lógica consistia simplesmente em um loop infinito, while(true), com várias alocações dinâmicas de memória , new.

Tendo criado o consumidor de memória, iniciamos a implementação do monitorador deste recurso, o MemoryMonitorLED/src/memoryPercentage.cpp. Nesta, decidimos usar uma biblioteca, a BlackGPIO.h, para ajudar no controle das portas GPIOs no momento de acender e apagar LEDs e também no acionamento do botão. Com esta biblioteca podemos instanciar uma porta GPIO da seguinte forma: BlackGPIO red(GPIO_44, output, FastMode), onde red é o nome do objeto, GPIO_44 é a porta escolhida e output configura a porta para ser usada como saída de dados. Após criar um objeto da classe BlackGPIO, é possível utilizar os métodos getValue e setValue, onde o último recebe como parâmetro um enum, low ou high, que representam os valores 0 e 1 respectivamente.

O próximo passo da implementação foi usar a função sysinfo da biblioteca sys/sysinfo.h para obter informações do sistema e a partir destas calcular o percentual de memória utilizado naquele momento.

Por fim, criamos um loop infinito, while(true), que calculava o percentual de memória a cada iteração e verificava em qual faixa este se encontrava. Se o percentual estivesse abaixo de 25% o LED verde acendia, se estivesse entre 25% e 50% o amarelo acendia, entre 50% e 75% o vermelho acendia e se fosse maior do que 75% todos os LEDs ficavam acesos. Neste último caso, se o botão fosse pressionado, o comando "ps aux --sort=-%mem | sed -n 2p | awk '{print $2}'" recuperava o PID do processo que estava consumindo maior quantidade de memória e o "kill -9 PID" era usado para matar este processo. A função sleep(1) foi usada para executar uma iteração do loop a cada 1 segundo.

2.2. COMO COMPILAR E EXECUTAR

Um makefile foi criado para ajudar da compilação do programa. Assim, para compilar o programa monitorador de memória, em um sistema Ubuntu, estando no diretório MemoryMonitorLED, basta digitar make no terminal. No

entanto, para usar esse makefile, é necessário ter instalado na máquina o compilador arm-linux-gnueabihf-g++ e suas dependências. Já o programa que consome memória de forma crescente pode ser compilado de forma simples, estando no diretório MemoryLeaker, como g++ memoryLeaker.cpp -o memoryLeaker. Após isso, o executável MemoryLeaker/memoryLeaker e o MemoryMonitorLED/bin/memoryLED gerado pela compilação do make podem ser enviados e executados na Beagle Bone Board. O monitoriador deve ser executado da seguinte maneira: sudo ./memoryLED.

2.3. RESULTADO

Um vídeo foi gravado com os programas em execução. Nele é possível ver o funcionamento dos LEDs e do botão. O vídeo pode ser acessado em https://drive.google.com/file/d/1PfPLLexi-LBlLm96k-iJzwVIOSDH766h/view?usp=sharing.

2.4. IMPLEMENTAÇÕES ADICIONAIS

Como implementação adicional poderia ter sido criada uma lógica para fazer todos os LEDs ficarem piscando ao ser atingido um percentual de uso de memória maior do que 75%. Além disso, ao ser pressionado o botão, poderia ter sido feita uma implementação mais aprofundada para decidir se era necessário matar o processo que estava consumindo maior quantidade de memória, ou seja, seria feita uma análise para saber se o consumo feito por tal processo era excessivo.