Pontifícia Universidade Católica do Paraná
Escola Politécnica
Bacharelado em Ciência da Computação / Cibersegurança

Curitiba, 23, maio de 2024.
Disciplina: Sistemas Operacionais Ciberfísicos
Professor: Jhonatan Geremias
Curso: Ciência da Computação
Nome Estudante: Gustavo Furini

TDE II - Trabalho Discente Efetivo

Smart Healthcare

Descrição da Atividade:

Esta atividade é composta por duas etapas. Primeiramente, responder as questões do item 1. Posteriormente, implementar uma solução de IoT destinada a *Smart Healthcare* no Contiki seguindo a especificação do roteiro.

Entrega:

Esta atividade deverá ser entregue até o dia **06/06/2022** no Canvas.

O estudante deverá entregar um arquivo ".pdf" contendo as respostas do roteiro de atividade item 1.

O item 2 corresponde a implementação do código no Contiki. O estudante deverá implementar o seu código apenas dentro do arquivo "socps.c". Entregar o arquivo "socps.c" onde foi codificado, este deve conter o nome do estudante e curso, adicionados no cabeçalho do arquivo em um bloco de comentário.

Ambos os arquivos devem ser postados no Canvas até a data limite da entrega.

Atenção: Favor alterar a extensão do arquivo "socps.c" para "socps.txt", os arquivos (.pdf e txt) devem ser postados individualmente, favor não compactar os arquivos.

Roteiro da Atividade

1. **Com apoio ao material fornecido responda:**
    a. Descreva o uso da macro PROCESS() no Contiki. Qual a função desta macro? Quais parâmetros ela recebe e para que são utilizados?

    R: No Contiki OS, um processo é declarado utilizando a macro PROCESS(). Essa macro permite definir um processo que será gerenciado pelo kernel do Contiki. Ela recebe dois parâmetros: o nome do processo e uma string que descreve o processo. O nome do processo é usado para o identificar dentro do sistema, enquanto a descrição serve para a depuração e monitoramento.

b. Descreva para que são utilizadas as macros PROCESS_BEGIN() e PROCESS_END() no Contiki?

R: As macros PROCESS_BEGIN() e PROCESS_END() no Contiki são utilizadas para delimitar o corpo de um processo. A macro PROCESS_BEGIN() marca o início do processo, configurando o ambiente necessário para sua execução e deve ser a primeira instrução na definição do processo. Já a macro PROCESS_END() indica o final do processo, encerrando sua definição e deve ser a última instrução. Essas macros são fundamentais para a correta estruturação e funcionamento dos processos no Contiki OS, garantindo que o código do processo seja executado no contexto apropriado.

c. O que faz a macro AUTOSTART_PROCESSES() no Contiki?

R: A macro AUTOSTART_PROCESSES() no Contiki é usada para especificar quais processos devem ser iniciados automaticamente quando o sistema é inicializado. Ela recebe uma lista de processos como parâmetros e garante que esses processos sejam automaticamente iniciados pelo sistema sem a necessidade de intervenção manual. Isso facilita a configuração inicial do sistema, garantindo que processos essenciais ou desejados estejam em execução desde o início.

2. Implemente o programa no Contiki conforme a especificação:

Contexto: Para auxiliar no combate da Pandemia do Covid-19, o centro pesquisa de equipamentos médicos está convidando você para participar do desenvolvimento de um novo projeto. O projeto consiste em monitor de dados vitais em um *smartwatch*. O monitor deve registrar os batimentos cardíacos do paciente (considerar entre 20 e 140 batimentos cardíacos - abaixo de 50 mensagem de batimento cardíaco baixo, acima de 90 apresentar mensagem batimento cardíaco alto), nível saturação do oxigênio (considerar oxigenação entre 80% e 100% - abaixo de 90% mensagem de saturação baixa) no sangue e temperatura (considerar temperatura de 34° a 41° - abaixo de 35° apresentar mensagem de hipotermia, acima de 37° mensagem de febre).

a. Implementar um programa no Contiki destinado a monitorar os dados vitais de uma pessoa (deve medir os batimentos cardíacos, saturação de oxigênio e febre);
b. Deverá ser criado três funções thread de processos, uma para monitorar cada um dos dados vitais;
c. Para simular os dados vitais deverão ser utilizadas funções randômicas;
d. Os dados vitais (aleatórios) devem ser gerados dentro de cada uma das funções thread de processo;
e. Imprimir os dados vitais em cada uma das funções;
f. Utilizar a função PROCESS_WAIT_EVENT_UNTIL() configurando um tempo de três segundos para cada função thread de processo.
g. Todas as funções thread de processos devem definir sua exclusão explicita utilizando a macro PROCESS_END();
h. Deverá ser criado uma quarta função thread de processo que deve aguardar por um evento;

i. Gerar um evento sempre que os dados vitais estiverem alterados (considerar os valores normais de cada um dos dados vitais);
j. O evento deve gerar uma mensagem de alerta para o usuário;
k. O código deve ser documentado, utilizar os comentários em toda a extensão do programa.