Processo seletivo | Estágio de Desenvolvimento de Software Embarcado III 2021

Leonardo Posso Benetti **email:** leonardo.benetti@usp.br

Este documento tem como objetivo expor minhas ideias e abordagens para resolver o case proposto no processo seletivo.

Primeiramente, comecei pelo modelo conceitual escrevendo um pequeno pseudocódigo de como eu gostaria que os programas funcionassem. **O programa em Python** foi o que gerou menos problemas, tanto por ser uma linguagem com a qual estou mais acostumado quanto pelo motivo das tarefas realizadas serem consideravelmente simples.

Este programa apenas pega seu ID de processo e o escreve em um arquivo de nome “pid”. A maneira como isso foi implementado faz com que este arquivo “pid” seja criado (ou atualizado) no mesmo diretório em que o script Python está rodando, mas fazer com que ele seja criado em outro diretório é trivial.

Depois deste arquivo ser criado, o programa percorre um loop que imprime “2: I am alive” por 3 vezes, aguardando x=2 segundos por iteração. Este tempo pode ser facilmente alterado ao mudar o valor da variável “x” no programa. Ao concluir este loop, ele imprime uma última mensagem e encerra sua execução. É importante notar que o ID de processo que foi escrito no arquivo “pid” continua lá, mesmo com o processo do programa python não estando ativo. Talvez fosse interessante removê-lo do arquivo ao final da execução, o que seria fácil de ser implementado, porém como isso não foi especificado no enunciado do case, não o fiz.

Neste programa Python implementei uma checagem básica de erros de execução. Julguei que os erros que poderiam ocorrer estavam ligados a abrir e escrever no arquivo “pid”, portanto coloquei essas operações dentro de blocos “try/except” para capturar eventuais exceções. Implementei de tal forma que se alguma exceção fosse gerada as informações sobre ela seriam armazenadas em um arquivo de nome “cromai.log”, com informações da data em que ela foi gerada e mensagens que podem ajudar a solucioná-la. Este arquivo, novamente, será gerado no mesmo diretório em que o script for executado e dados novos serão colocados ao final dele. Para gerar um arquivo de log, pode-se, por exemplo, cometer um erro de sintaxe proposital dentro do bloco “try/except” e observar o arquivo gerado.

O programa em Shell é relativamente simples, porém foi o que gerou mais mudanças do modelo conceitual. Ele é composto por um loop infinito que lê o arquivo “pid” e testa se há algum programa Python ativo com o ID lido.

Para realizar esse teste é executado o comando:

“$(pgrep python -d ' ')”

Ele obtém o ID de todos os programas python sendo executados, sendo separados por um espaço (isto é, delimitador ‘ ‘).

Tendo este resultado na variável “py_procs” e o ID lido do arquivo “pid” na variável “pid”, fiz o teste condicional:

if [["$py_procs" == *"$pid"*]]

Este teste identifica se o ID lido no arquivo “pid” está dentro da string de todos os ID’s de programas Python sendo executados. Caso esteja, significa que o programa Python está ativo e o script shell imprimirá “1: It is alive”. Senão, ele não está ativo e será impresso “1: It is dead”.

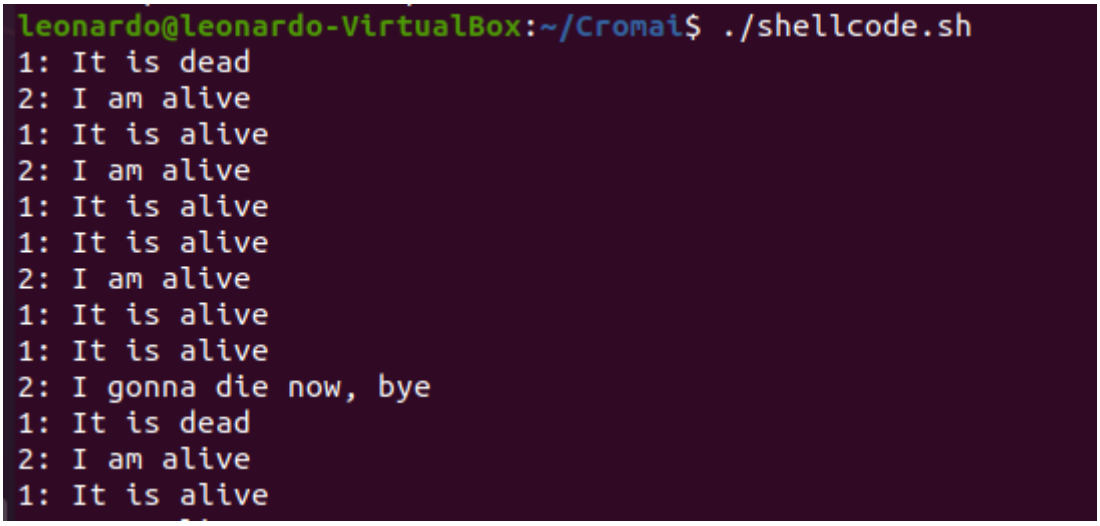
Outra mudança do modelo conceitual é que entendi primeiramente que o programa Python deveria ser chamado a cada iteração do loop infinito. Isso, entretanto, faria com que vários programas Python fossem executados concorrentemente, sempre atualizando o arquivo “pid”. Julguei que isso não é a proposta deste case. Para mim, faz mais sentido que o programa só seja chamado quando o script Shell identifica que ele não está em execução.

Em resumo, essa mudança significa que o programa Python será chamado dentro do “else” do teste condicional, e não fora do teste em toda iteração. Assim apenas um programa estará ativo por vez. Se essa não fosse a ideia original do case, basta mudar a linha 20 “python3 pythonscript.py &” para a linha 22, fora do “else”.

Outra pequena mudança implementada é que adicionei um “sleep 1” no final do loop infinito, para que suas iterações ocorram um pouco mais devagar e fosse possível observar melhor o resultado. Isso não é essencial para o funcionamento, portanto pode ser retirado do programa sem grandes problemas.

A seguir pode-se observar uma imagem da execução do programa Shell integrado com o Python. Nota-se que no primeiro momento o Shell não observa programa Python ativo e imprime “1: It is dead” e chama a execução do programa Python. Este, por sua vez, atualiza o arquivo “pid” e imprime “2: I am alive”. Logo, na próxima iteração ao ler o arquivo “pid” novamente o programa shell identifica que o programa Python está ativo e por isso imprime “1: It is alive”.

Isso se repete até que a execução do programa Python acaba, sinalizada pelo “2: I gonna die now, bye”, o que retorna o sistema para o estado inicial, repetindo todo esse procedimento.



Afirmo aqui que todos os programas desenvolvidos são de minha autoria e me coloco à disposição para eventuais dúvidas.