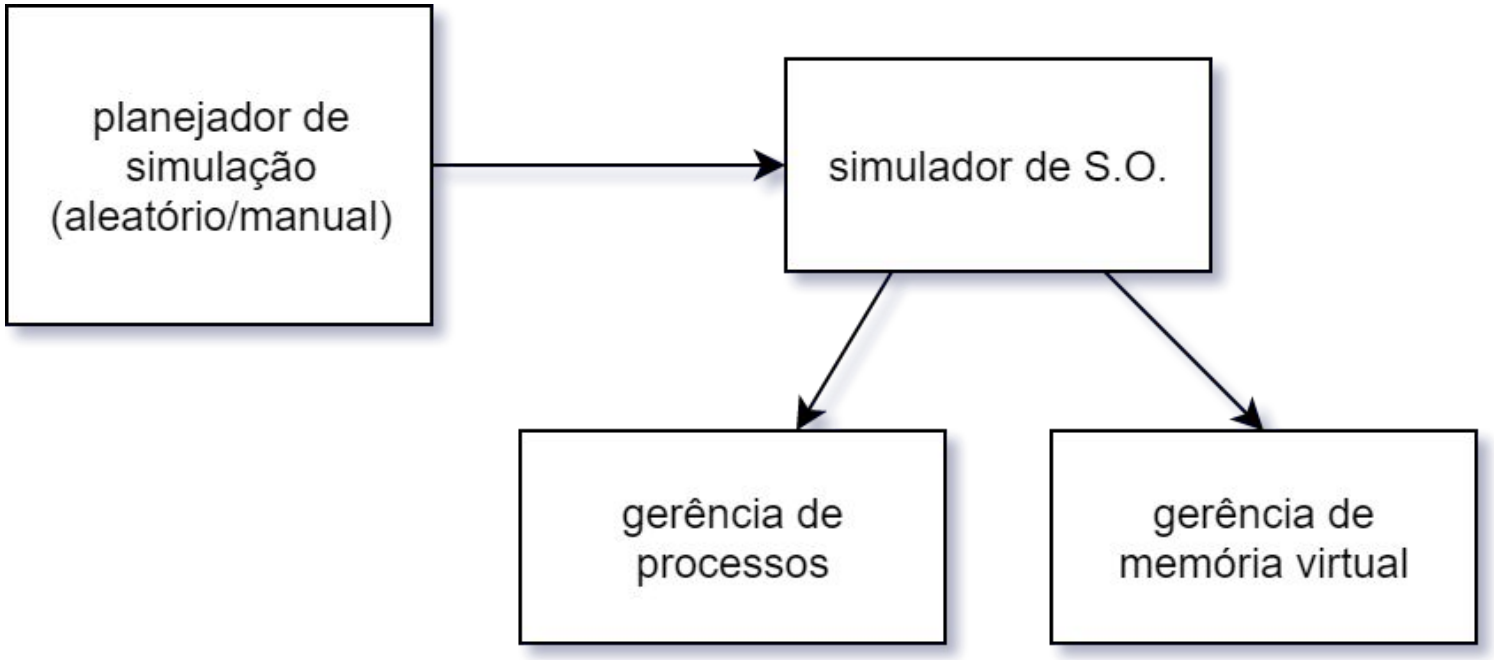
Sistemas Operacionais

Miguel Angelo
Erick Rocha

Overview


planejador de simulação (aleatório/manual)
simulador de S.O.
gerência de processos
gerência de memória virtual

Estruturação geral

- Separação dos módulos:
  - simulador de S.O. (parte interna do simulador)
  - planejador/driver de simulação (parte externa ao simulador)
- Dois drivers implementados:
  - manual: simula processos e requisições de memória a partir de arquivo txt
  - aleatório: simula processos e requisições de memória de forma aleatória
- Estruturas internas do simulador feitas na mão:
  - map e hash
  - queue
- Estruturas externas ao simulador:
  - usamos bibliotecas de terceiros: uthash, utarray, pgc32
  - implementamos: distribuições aleatórias - uniforme e poisson

Código fonte

- Estruturas necessárias para o planejamento e gerenciamento da simulação
  - sim_plan.h
- Componentes de simulação
  - aleatória: sim_plan_rand.h // .c
  - manual: sim_plan_txt.h // .c
- Procedimentos da simulação
  - plan_txt_create_device: simulação da criação de um device (e.g. IO).
  - plan_rand_init: inicialização do planejamento da simulação.
  - plan_rand_incoming_processes: simulação de chegada de processos.
  - plan_rand_create_process: simulação da natureza do processo a ser criado.
  - plan_rand_execute_memory: simulação da seleção das páginas a serem executadas.

Escalonador

- Loop de Simulação
  - Lógica que ocorre no início de cada unidade de tempo
    - Simulação de chegada de processos
    - “Promoção” de processos nas filas de espera
    - Checagem dos IO devices por processos terminados
    - Lógica de preempção (swap-in // swap-out)
    - Lógica de gerenciamento de page-faults
    - Lógica de gerenciamento de operação de IO
    - Lógica de término de processos (dispose)
  - Após a execução da unidade de tempo
    - impressão no console do resultado da unidade de tempo

Gerenciamento de Memória

- Aplicação, em diversos componentes do OS, de conceitos de gerenciamento de memória
  - Gerenciamento de frames de memória disponíveis
    - Estratégia LRU
    - Lógica de seleção de frames livres para carregar página(s):
      - Pode gerar “ERR_OUT_OF_MEMORY”
    - Lógica de gerenciamento de páginas:
      - Pode gerar “ERR_PAGE_FAULT”
  - Page Ready Queue
    - Fila para executar processos após o swap-in com alta prioridade

Limitações

- Não há checagens após invocar malloc
- Todas as filas suportam o máx de processos, isso não é realista
- Usamos time-stamp para indicar os frames mais recentemente usados
- Fila de espera de frame deveria ser limitada para impedir excesso de swap-out
- Memória instalada é definida em termos de frames e não de bytes
- Tamanho da área swap é indefinidamente grande
- Entry point de qualquer processo é o endereço virtual 0 (zero)
- A simulação não irá forçar um limite na quantidade de memória virtual endereçável

Limitações

- O design é monolítico, misturando gerência de CPU com memória
- Não há simulação do translation lookaside buffer
- Cada processo tem uma tabela de páginas, ou seja, não é tabela de páginas invertida
- Swap in/ou não sabe o endereço da página dentro do storage
- Há apenas uma CPU, não há simulação de multiprocessamento
- Cada página tem 4KB
- Não há threads, a unidade é o processo
- O sistema operacional reserva as 5 primeiras páginas para si mesmo

Conclusão

- Tivemos problemas com alocação e liberação de memória
  - usamos uma ferramenta especializada para achar o problema
- Problema com loop infinito no swap-out
- Swap-in não foi implementado
- Gerência de tempo de CPU funcionando

Conclusão

- Gerência de estado do programa é complicada
  - o estado do processo tem que ser lembrado para continuar a operação após swap-in/out
  - exemplo:
    - P1 quer executar o endereço na página G1
    - requisição de memória do endereço G1 a ser executado
    - G1 não está presente, gerando page-fault
    - carregar página G1 do processo P1 do swap
    - não tem memória
    - pedir swap-out de outro processo P2
    - escrever páginas de P2 no disco (swap-out do processo P2)
    - quando acabar de escrever, continuar carregamento de página do swap para P1
    - quando acabar carregar, continuar executando P1

Compilação

Na raiz do projeto execute:

```
gcc -I src/libs -I src/os src/os/*.c src/libs/*.c src/*.c -o a -g
```

Se for no Windows, basta adicionar em .exe ao nome do arquivo compilado:

```
gcc -I src/libs -I src/os src/os/*.c src/libs/*.c src/*.c -o a.exe -g
```

-g é para inserir informações de debug, sem isso não é possível debugar

Libs utilizadas

- Internas do SO
  - hash.h //.c
  - map.h //.c
  - math_utils.h //.c
  - queue.h //.c
  - return_codes.h
  - safe_alloc.h //.c
- Externas ao SO:
  - rand_distributions.h //.c
  - ansi_colors.h
  - ansi_console.h
  - rand.h //.c - 3rd party
  - utarray.h - 3rd party
  - uthash.h - 3rd party

Referências

- Site da disciplina de S.O. da professora Valéria - https://dcc.ufrj.br/~valeriab/mab366.php
- Page table e frame table: https://en.wikipedia.org/wiki/Page_table
- LRU e marcação de páginas - https://en.wikipedia.org/wiki/Page_replacement_algorithm#Least_recently_used
- Process Swapping in Minix by John Homer - B. S., Harding University, 2002 - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1877&rep=rep1&type=pdf

Referências

- Generating numbers
  - Geração de números aleatórios - https://en.wikipedia.org/wiki/Permuted_congruential_generator
    - https://stats.stackexchange.com/questions/337927/is-pcg-random-number-generator-as-good-as-claimed
  - PCG32 - http://www.pcg-random.org/
  - Distribuição poisson - https://en.wikipedia.org/wiki/Poisson_distribution#Generating_Poisson-distributed_random_variables
  - Random numbers - https://www.random.org/
  - Prime numbers - https://www.dcode.fr/next-prime-number
- Data Structures
  - uthash - http://troydhanson.github.io/uthash/index.html
  - utarray - https://troydhanson.github.io/uthash/utarray.html
- Compiladores, debuggers e outras ferramentas utilizadas
  - GCC/GDB - https://gcc.gnu.org/
  - Application Verifier - https://docs.microsoft.com/pt-br/windows/win32/win7appqual/application-verifier
  - VSCode - https://code.visualstudio.com/