Universidade de Brasília
Aline Laureano de Araújo Vilela
Helena Bretas Goulart
Sara Conceição Silva

Gerenciamento de memória em sistemas operacionais

Projeto de simulação de memória e manipulação de processos e arquivos através da estrutura de dados e algoritmos na linguagem C.
Docente: Fernando William Cruz.

Brasília – DF
2017

1.0 - Objetivos

A estrutura de dados e algoritmos contempla o conhecimento sobre alocação de memória e a manipulação de dados através de vetores, listas, filas, pilhas e árvores. O projeto de gerenciamento aborda uma simulação prática do uso de memória através da criação, inserção e remoção de processos em listas circulares duplamente encadeadas, tratando também dos espaços de memória disponíveis, chamados “holes”, e como otimizá-los de modo que os pequenos espaços tornem-se um e formem o maior tamanho de memória possível, possibilitando que caibam processos maiores ou maior quantidade de pequenos processos. Os processos possuem dados visíveis para o usuário do programa: é possível ver em tempo real o horário em que eles foram criados e encerrados, assim como o tempo de execução e o seu tamanho.

2.0 – Diário de atividades

07/10/2017
Através de uma conversa online via Hangouts, a primeira reunião do grupo teve seu início com a leitura em conjunto das dicas e requisitos do projeto, pontuando previamente como eles poderiam ser resolvidos e quais estruturas e funções poderiam ser utilizados em cada caso. Houve um breve estudo sobre a função *rand* e sobre a criação de arquivos em linguagem c. Nesse mesmo dia foi feito o início do código com a criação do cabeçalho, dos nós e da função de inserção no início da lista, e surgiram dúvidas sobre como seria definido o tamanho do processo, se pelo leitor ou através de um número randômico.

08/10/2017
A programação em pares realizada pelo compartilhamento de telas no Hangouts possibilitou a compilação do que havia sido escrito até então e a correção dos erros e avisos do tipo “warning”. Houve também uma revisão do código para conferir se ele atendia ao critério de lista circular duplamente encadeada e a criação do menu com as opções de “switch case” para a melhor orientação das demais funções que deveriam ser criadas.

09/10/2017
A discussão foi também através do Hangouts, na qual foi decidido não evoluir com o código devido a dificuldade em fazer casos gerais para a inserção de processos na memória, uma vez que o código estava com muitas especificações para cada possível caso.

10/10/2017

As dúvidas referentes a solicitação da memória e salvar os processos em arquivo foram sanadas durante a aula com o professor Fernando William. Uma reunião pessoalmente foi realizada posteriormente para que a função de inserir processos fosse finalizada, mas novamente não foi encontrada uma forma de generalizar os casos e foi necessária a ajuda dos monitores da disciplina que indicaram git cakren para a realização do projeto, incentivando também a pesquisa de de funções que captam a hora atual do sistema.

11/10/2017
Nesta data concluiu-se a inviabilidade do uso de git craken, que demandaria tempo até que todas se habituarem ao programa. A função de inserção ainda não havia sido concluída e a maior dificuldade permanecia sendo como inserir um novo processo devido às possíveis combinações distintas de espaços de processos e memória disponível.

12/10/2017
A reunião do grupo ocorreu durante todo o dia, e a melhor alternativa encontrada fora especificar os casos de inserção e generalizar quando possível. Logo após tornou-se simples o desenvolvimento do código, tendo em vista que a lógica completa do programa já havia sido discutida repetidas vezes. Sendo assim, as funções de remoção de processos, verificação de tempo dos processos, imprimir dados e salvá-los em arquivo e foram criadas e o desenvolvimento do código estagnou-se quando o programa deixou de unir buracos para transformá-los em espaços de memória maiores para serem utilizados.

13/10/2017
O dia da entrega do projeto foi separado para testes e correção de erros. Os principais erros encontrados foram na função de união de espaços vazios, o que possibilitou encontrar um erro que estava impossibilitando o comportamento ideal das demais funções: o número randômico apresentava valores muito baixos em questão de segundos, e algumas funções não possuíam tempo sequer de serem executadas. O erro logo foi corrigido através da definição de um tempo mínimo e um tempo máximo para os processos

3.0 - Requisitos atendidos

- Os processos possuem tempo de execução, tamanho e um label identificador;
- Os processos estão sendo inseridos na lista(memória) de acordo com seu tamanho.
- Os processos estão sendo inseridos no primeiro lugar onde encontram espaço suficiente.
- No caso de um processo não caber nos buracos de memória disponíveis, o programa encontrou uma forma de reorganização, unindo os buracos disponíveis na

memória para acomodar o novo processo, porém a função de compactar memória não foi conferida adequadamente.
- Se o processo não couber na memória, o programa informa o usuário.
- O tempo de execução de um processo é aleatório, porém, colocamos limites mínimo e máximos para que tenhamos tempo suficiente para fazer os testes das funções. O grupo decidiu simular esse tempo através da função rand().
- O processo termina quando seu tempo acaba e também quando o usuário pede.
- O programa tem a opção de imprimir os dados do processo.
- O programa é capaz de imprimir, a qualquer momento, quais posições de memória estão ocupadas (P) e quais estão livres (H).
- O programa tem uma opção de sair (deixar o programa) e oferece a opção de gravar os dados da memória em arquivo.
- A gerência de espaços de processo (P) e buracos (H) na memória foi feita com o uso de uma lista circular duplamente encadeada, com cabeçalho. E cada nó possui um tipo(H ou P).

4.0 - Falhas e limitações

Nao terminamos a função de usar as informações do arquivo para continuar gerenciando a memoria.

5.0 - Opiniões pessoais

5.1 Aline Laureano de Araújo Vilela

O projeto apresentou uma lógica interessante e de difícil aplicação, por isso julguei como essencial que a parte prática começasse o quanto antes. A função de inserção apresentou grande bloqueio do código devido às diferentes possibilidades que poderiam existir na organização de processos e memórias, mas possibilitou o maior aprendizado do conteúdo e facilitou que as demais funções do código fossem criadas. A parte mais difícil foi, sem dúvidas, consertar os erros de execução do programa tais como solicitar duas vezes seguidas a mesma opção do menu e união de buracos na memória.

5.2 Helena Bretas Goulart

O projeto possui implementação complexa, e as dificuldades iniciais se basearam no uso de funções das quais não conhecia antes, como rand e time.h. A lógica do programa não apresenta um nível de dificuldade tão grande quanto sua implementação, sendo que esta exige a atualização de ponteiros e structs, uma vez que o requisito de ser lista circular duplamente encadeada precisava ser atendido. Acredito que implementar esse código sozinha não seria possível, e todos os membros do grupo trabalharam em conjunto para que pudesse ser concluído. A maior dificuldade ocorreu quando não houveram erros de compilação, e sim erros

de execução da lógica desejada - e para solucionar foi necessário acrescentar “printf” em diversos laços para verificar se as condições estavam sendo processadas ou não.

5.3 Sara Conceição Silva

A maior dificuldade encontrada foi em como implementar as funções de maneira que atendesse todos os casos sugeridos nas instruções do projeto e atender aos requisitos. O projeto em si, não tem uma lógica muito difícil, porque depois que desenhamos a lista e os casos de teste num papel, percebemos quais seriam nossas dificuldades na implementação. Depois as dificuldades foram os erros de execução, já superados os erros de compilação. E também para usar o arquivo salvo.