Relatório – Trabalho de EDA (13/10)

Estruturas de Dados e Algoritmos (EDA) – Turma: A

Integrantes:

Arthur Evangelista dos Santos – 140016686

Rafael Makaha Gomes Ferreira -- 160142369

Renan Welz Schadt -- 160143403

Objetivos do Programa:

O objetivo do programa é simular a memória RAM de um sistema operacional. O usuário insere o tamanho total da memória a ser simulada, e insere o nome e o tamanho dos processos a serem colocados nela. A gerência dos processos e lacunas é feita através de uma lista circular duplamente encadeada.

Os processos têm uma duração definida “aleatoriamente” em um intervalo de 2 a 1000 segundos (intervalo este que pode ser facilmente modificado), após esse tempo expirar, ele deve trocar seu tipo para buraco.

O programa também deve ser capaz de unir lacunas adjacentes e reorganizar a lista caso um processo não se encaixe em nenhuma das lacunas disponíveis Nesse caso, ele junta todos os espaços em um só“slot” no final da lista.

O usuário pode imprimir a lista, exibindo os processos em execução e os espaços vazios quando desejar.

Ele também tem a opção de salvar o estado atual da lista em arquivo e inicializar o programa novamente através deste.

Todas funções especificadas acima foram implementadas e funcionam conforme o esperado.

Análise de Falhas e Possíveis Melhorias:

O programa contém algumas limitações. Entre elas é possível citar que se dois processos forem inseridos no mesmo segundo, eles terão a mesma duração, devido a semente do comando “srand” da biblioteca <time.h> ser o tempo atual do computador em segundos.

Para evitar o problema citado anteriormente pelo comando srand poderia ser utilizada a função delay(). Porém, ela interromperia o sistema para execução deste delay. Outro problema de se utilizar esta função é seu parâmetro que equivale a milissegundos, sem que se tome em conta o tempo de processamento do computador para contabilizar cada milissegundo. Por estes motivos, foi dada a preferência pela função srand() no programa final.

Também é possível afirmar que o programa só se atualiza e remove os processos expirados quando o usuário digita algo, o scanf pausa o programa, porém isso é imperceptível para o usuário, assim que o scanf é executado o programa imediatamente checa se tem algum processo expirado. É importante observar que a limpeza do buffer do teclado a cada momento de leitura de uma string pode solucionar possíveis falhas na inserção de labels de processos.

Existem funções “scan” que não pausam o programa quando são chamadas, que poderiam ser implementadas, caso o scanf tradicional não fosse suficiente para alcançar o objetivo do programa.

Outra limitação é que o tamanho dos processos deve ser do tipo variável inteira. Caso seja inserido um valor do tipo float ou double não serão consideradas as casas decimais.

Usar uma semente diferente para permitir inserções no mesmo segundo.

Usar e suportar o tipo double para armazenar o tamanho do processo.

Diário de Atividades:

A organização do diário de atividades teve como base as funções a serem implementadas no programa. A data que elas foram implementadas e seu autor/autores constatados a seguir. Nossa estratégia foi programar em ao menos duas pessoas, juntas em uma só máquina ou através de videoconferências. Esta estratégia proporcionou uma redução no número de erros e um processo de “debug” mais rápido. O uso do programa GNU Debugger (GDB) também auxiliou o processo por permitir o debug linha a linha do código. Bugs mais difíceis de serem observados foram corrigidos de maneira relativamente mais rápida quando comparados a ausência da ferramenta.

07/10 - Reunião entre os integrantes para discutir o projeto e definir as estruturas de dados a serem utilizadas no programa. Foi criado um Checklist de requisitos a serem especificados para manter controle dos requisitos já implementados ou implementados parcialmente. Ainda neste dia, cada integrante realizou um estudo preliminar dos conceitos envolvidos no livro "Sistemas Operacionais Modernos" de Andrew S. Tanenbaum.

09/10 - As structs foram definidas e foi feita a estrutura básica da main, as funções implementadas nesse dia foram:

Cab* iniciaCab(Cab* l, int mem_total); **(Rafael e Renan)**

int lerMemoriaTotal(); **(Rafael e Renan)**

void lerNome(char* s); **(Rafael e Renan)**

void menu2(Cab* l); (estrutura básica inicial) **(Rafael e Renan)**

10/10 - As funções implementadas foram:

int checaMemoriaLivre(Cab* l, int temp_tam); **(Renan)**

int inserirProcesso(Cab* l, char* nome, int tamanho); **(Arthur, Rafael e Renan)**

Houve uma certa dificuldade na implementação da função inserirProcesso, pelas suas várias exceções, como inserção no começo, fim, inserção do primeiro elemento, entre outros.

11/10 - As funções implementadas foram:

int numeroRandom(); **(Arthur e Rafael)**

int retornaTempo(); **(Rafael)**

void checaTempo(Cab* l); **(Rafael e Renan)**

Processo* removerProcesso(Cab* l, Processo* p); **(Rafael e Renan)**

Processo* unirBuracos(Cab* l, Processo* p); **(Rafael e Renan)**

void imprimirProcessos(Cab* l); **(Arthur)**

Usamos a biblioteca “time.h” para introduzir número randômico e verificar a duração dos processos, tentamos fazer utilizando a biblioteca “stdlib.h”, mas ela só armazena o horário no momento em que o programa foi compilado.

A dificuldade em unirBuracos foi a mesma de inserirProcesso, os vários casos.

A dificuldade em checaTempo e removerProcessos foi acertar os momentos onde chamar a função, para que ela possa estar checando a duração e removendo os processos expirados o mais cedo possível.

12/10 - As funções implementadas foram:

void realocarMemoria(Cab* l); **(Arthur, Rafael e Renan)**

void salva_arquivo(Cab* l); **(Arthur)**

Cab* carrega_arquivo(Cab* l); **(Arthur)**

void escreverProcesso(Cab* l,Processo* aux); **(Arthur e Rafael)**

int menu1(Cab* l); **(Arthur)**

A função realocarMemoria foi a que tomou mais tempo para ser implementada, com problemas de double free quando compilada, a função foi refeita algumas vezes, funcionando de maneira um pouco diferente em cada, alguns erros passaram despercebidos nas implementações anteriores, a última implementação funciona conforme o esperado.

Houve também dificuldades na função carrega_arquivo, devido as variáveis, inicialmente, não serem salvas corretamente e pela falta de prática com as funções voltadas a manipulação de arquivos.

13/10 - Nenhuma função foi implementada, comentamos o código e testamos as funções corrigindo os últimos erros restantes.

Checklist de Requisitos

Itens marcados com um X indicam itens concluídos até o momento da entrega deste relatório.

- [X] Implementar lista circular duplamente encadeada com cabeçalho
- [X] Modularizar lista implementada
- [X] Inserir ENTRADA de parâmetros (processo: tamanho e tempo de execução)

- [X] Implementar menu com opções
- [X] Inserir label de identificação do processo
- [X] Requisito de alocação (*De acordo com o tamanho)*
- [X] Requisito de alocação (*Do início para o final)*
- [X] Exceção de alocação (*Reorganizar lista)*
- [X] Exceção de alocação (*Não foi possível alocar)*
- [X] Encontrar solução para simular tempo de execução e término do processo
- [X] Implementar solução p/ tempo de execução
- [X] Implementar printf com registro de entrada e saída
- [X] Implementar printf com áreas ocupadas (P) e livres (H)
- [X] Inserir opção SAÍDA de parâmetros
- [X] Inserir opção SAIR DO PROGRAMA
- [X] Inserir opção de inicialização "*Deseja continuar simulação?"*
- [X] Implementar abrir simulação em arquivo
- [X] Inserir opção de finalização "*Deseja salvar simulação em ARQUIVO?"*
- [X] Implementar gravar/ler simulação em arquivo

Opiniões Pessoais Sobre o Projeto:

Arthur Evangelista dos Santos:

Projetos deste porte são importantes para formação acadêmica do estudante universitário. O planejamento de uso dos recursos, tempo e gerenciamento do projeto como um todo ensinam muito. Talvez o recurso com maior demanda foi o tempo, tendo em vista a rotina atribulada dos integrantes e, possivelmente, dos demais colegas de turma. A dinâmica do grupo e a boa comunicação entre os integrantes foi de suma importância. A implementação de diversas funções escritas e modularizadas por indivíduos diferentes, com lógicas e práticas de programação distintas é um desafio interessante.

Rafael Makaha Gomes Ferreira:

O projeto foi de grande ajuda para a aplicação de listas encadeadas e revisão de manipulação de arquivos em linguagem C. Com uma certa medida de dificuldade, porém vencida com esforço e dedicação pessoais de cada indivíduo do grupo. Houveram dificuldades mais na parte de implementação do que na abstração das pequenas partes inicialmente separadas. O tempo disponível para a realização do projeto foi de

grande dificuldade para organizar devido a grande demanda de tempo de outras disciplinas proporcionadas pela faculdade.

Renan Welz Schadt:

Achei o projeto muito interessante, o fato de implementar um “mini” sistema de gerenciamento de memória, ajuda a compreender melhor como os sistemas operacionais funcionam. A implementação tomou muito tempo, às vezes um erro passava despercebido e demoramos a encontrá-lo. O grupo teve que aprender as funções relacionadas a contagem de tempo e praticamos muito a implementação de listas encadeadas circulares. O maior empecilho foi o tempo limitado para entrega.