SISTEMA PREPARADOR DE TEXTOS E PROGRAMAS

Paulo Mário Bianchi França

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.)

Aprovada por:

Ivan da Costa Marques

Ysmar Vianna e Silva Filho

Tercio Pacitti

RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

JUNHO DE 1974

Ā TERESA

AGRADECIMENTO

A todos os funcionários do NCE tanto pelo apoio em nosso trabalho quanto pelo simples companheirismo; em especial ao orientador prof. Ivan da Costa Marques, ao prof. Ysmar Vianna e Silva Filho, e ainda aos colegas: Paulo Fernando Molo Paiva, Newton Faller, Laerte Garcia Martins e Pedro Manoel Silveira que continuam envolvidos na implementação do projeto.

É ainda oportuno ressaltar que o presente projeto foi parcialmente financiado pelo Conselho de Pesquisas para Graduados (CEPG-UFRJ).

RESUMO

O presente trabalho consiste no projeto de um sistema preparador de programas que visa facilitar o trabalho dos usuários de computador no preparo e submissão de seus programas.

Tendo em vista tratar-se de um sistema ao mesmo tempo mais poderoso e econômico do que as populares perfuradoras de cartão, seu desenvolvimento visou a substituição futura deste equipamento.

O sistema consiste do software para um minicomputador acoplado a diversos terminais e em ligação local ou remota com outro computador onde os programas são processados.

Deve ser ressaltada uma diferença fundamental com os sistemas de "time-sharing" que em nada competem com o sistema apresentado pois são voltados para oferecer muito mais potencialidade porém a um número relativamente restrito de usuários e com custo absolutamente não-competitivo com as atuais perfuradoras. Ao contrário, êste sistema é destinado a atender a uma massa de usuários com custo inferior ao das perfuradoras.

ABSTRACT

This work is the design of a Program-Entry System which is expected to help computer users in making and submitting their programs.

Since this system is more economical and powerful than the still very popular keypunches, its development was made having in mind the future replacement of these.

The system consists of the complete software for a minicomputer connected to many terminals and attached either local or remotely to another computer where the user programs are to be executed.

It should be noted that this system is fundamentally different from Time-sharing systems since these are aimed to provide much more facilities to a relatively restrict number of users at a cost absolutely non competitive with traditional keypunches. Instead, the present system was designed to serve a wide number of users at a cost lower than that of keypunches.

ÍNDICE

I. INTRODUÇÃO 1

Motivações do projeto 1

Equipamento e configuração 7

Introdução ao Sistema 8

Premissas 10

Definições Iniciais 12

II. COMANDOS DO SISTEMA 13

Procedimentos 13

Comandos do Sistema 14

Identificação do operador 24

III. TABELAS DO SISTEMA

Tabela de Informação de Terminais (TIT) 29

Tabela de Disco 31

Limbo 32

Tabela de Informação de Cassettes 34

Tabela de Reservas 35

IV. ORGANIZAÇÃO DO DISCO 36

Características Técnicas 36

Organização Lógica 37

Acesso a linhas de Texto 39

Alocação de Blocos e Setores 39

Algoritmo de Estouro 45

Arquivo de Contrôle 49

Diretório 51

Registro de Rótulo 55

Administração da memória 61

V. MONITORAMENTO 66

Funcionamento 68

Mudança de contrôle 71

I. INTRODUÇÃO

MOTIVAÇÕES DO PROJETO

A escolha do projeto em questão foi feita observando-se a linha de pesquisa seguida pelo grupo de computação da UFRJ e a justificativa econômica adiante descrita.

A linha de pesquisa adotada pelo grupo de computação da UFRJ tem como um dos seus pontos fundamentais o desenvolvimento de projetos visando a resolução de problemas locais (encontrados dentro da UFRJ), porém de interesse regional ou nacional de modo a possibilitar a industrialização do produto. Dessa maneira se contribuirá para que um objetivo mais importante,qual seja o desenvolvimento da indústria de computação no país, seja atingido.

Respeitando esta política foi concebido o sistema PRETEXTO visando resolver o problema de preparação de programas por alunos e pesquisadores na UFRJ.

A computação ou processamento de dados em uma universidade tem características próprias que a distingue essencialmente do processamento de dados na indústria ou no comércio. Entre os parâmetros que, na nossa opinião, são indicativos destas diferenças estão:

- número de usuários: como consequência da rápida difusão do computador como uma versátil ferramenta de trabalho, em uma universidade moderna o número de usuários do computador tende a ser uma fração substancial do número total de alunos e pro

fessores da universidade. Isto cria o problema de se dar acesso à máquina para uma grande massa de pessoas e simultaneamente se manter economicamente um tempo de resposta aceitável.

- diversidade do nível de treinamento dos usuários: não se pode esperar uniformidade de treinamento ou conhecimento dos usuários na universidade. Pela própria finalidade da instituição, dentro da universidade vamos sempre encontrar, ao lado de pessoas que têm um conhecimento de vanguarda na área de computação, principiantes que estão tendo seu primeiro contato com a máquina e com os conceitos básicos de computação. Isto cria o problema de se dar ao usuário o tipo de acesso à máquina mais adequado ao seu nível de conhecimento.

- diversidade de tarefas: no ambiente universitário cada usuário vai ao computador para resolver seu próprio problema, desde o aluno principiante que está fazendo um curso de introdução à computação, até uma equipe de pesquisa que pode estar construíndo todo um sistema computacional. Isto cria o problema de que a máquina deve ser oferecida aos usuários sempre em nível de programação, isto é, o usuário típico na universidade vai ele próprio programar a máquina, ao invés de fazer simplesmente uma entrada de dados para um programa existente no sistema. Um outro problema criado pela gran

de diversidade das tarefas é a impossibilidade de se manter, sob qualquer forma, em arquivos do próprio sistema todos os programas de todos os usuários. Portanto, o usuário típico na universidade tem que manter seus próprios programas de alguma forma fora do sistema; seja em cartão perfurado, fita de papel perfurada, fita magnética ou alguma outra forma de armazenamento que possa servir como entrada para o sistema.

- existência de programas curtos: em todo centro de processamento de universidade uma parte substancial do trabalho é dedicada aos chamados programas curtos. Tais programas são os exercícios dos cursos básicos de computação ou de matérias tais como cálculo numérico, em que os alunos recebem como tarefa, normalmente semanal ou bisemanal, a confecção de pequenos programas. Estes programas curtos têm como característica principal o fato de que quase todo o seu tempo de processamento é compilação, pois além de normalmente não envolverem muita execução ainda são na maioria dos casos compilados muitas vezes e executados uma só vez. A existência destes programas curtos indica que um grande esforço, da parte de toda uma massa de usuários, é colocado na confecção de programas fontes. Esta situação nos indica que o sistema de computação universitário deve prover para uma grande massa de usuários métodos de acesso orientados para a compilação de programas e para a deteção rápida de certos erros nos programas fonte.

Um dos grandes problemas de computação na UFRJ e no ambiente universitário em geral é a perfuração. Atualmente, só nos cursos de Computação I estão matriculados cerca de 1.200 alunos por semestre, além dos alunos que, mensalmente, frequentam cursos intesívos do NCE e outros tantos alunos de graduação e pós-graduação usuários do computador.

A solução usualmente adotada é a compra de perfuradoras. Apesar de que a entrada de dados por cartão seja criticada como "obsoleta" as alternativas são difíceis para o ambiente universitário. Isto porque a demanda de perfuração é dedicada quase que totalmente à confecção de programas que são constantemente alterados e consertados. Daí a comodidade do cartão continuar prevalecendo sobre outras características oferecidas por sistema - de entrada de dados.

De um modo geral, as alternativas disponíveis no mercado são as seguintes:

a) Sistema de Time-Sharing ou Preparadores de Programas (editores de texto) on-line

Tais sistemas podem substituir com muita vantagem (para o usuário) as perfuradoras. Têm, no entanto, como incoveniente seu alto custo uma vez que necessitam de um computador de grande porte para atender a um número não muito alto de terminais.

Esta solução seria portanto inadequada a UFRJ e, certamente proibitiva a universidades de menor porte. (Note-se que o uso de tais sistemas não está impugnado; apenas seu uso como substituição de perfuradoras).

O custo dos sistemas "time-sharing" são proibitivos para uma

massa de mais de 1.000 alunos. Isto porque, devido a sua natureza interativa, o tempo de ocupação de cada terminal por usuário é muito superior ao tempo de ocupação de uma perfuradora pelo mesmo usuário, resultando - num investimento equipamento/usuário muito grande.

b) Sistema de entrada de dados (data-entry)

Tais sistemas consistem de terminais inteligentes, ou terminais ligados a mini-computadores (em alguns casos encontra-se terminais ligados a computadores de maior porte) e, possibilitam aos usuários certas vantagens normalmente não oferecidas pelas perfuradoras tais como: crítica dos dados, acêrto etc.

Os dados entrados dessa forma são gravados em um meio de transferência (normalmente fita magnética) e, posteriormente, alimentados no computador.

Embora usados com sucesso em instalações comerciais, êstes sistemas não são apropriados à preparação de programas porque não oferecem facilidade de se inserir ou alterar registros em locais apropriados, que constitui o principal problema do programador pois durante a depuração o programa necessariamente sofre diversas alterações.

O custo dêstes sistemas normalmente regula com o das perfuradoras no sentido de que o custo por terminal é aproximadamente o custo de uma perfuradora. Devido a sua melhor performance, no entanto seu uso é mais econômico.

Apesar de que, para a finalidade proposta, não haja um substituto ideal, o custo de perfuradoras, é muito elevado, agravado, recentemente, pela substituição no mercado das perfuradoras IBM 029 por IBM 129.

Assim sendo, o sistema PRETEXTO foi concebido com algumas características de data-entry (usa um minicomputador) porém oferecendo um editor de textos para maior facilidade no preparo dos programas.

Note-se que, tendo em vista a abertura de um mercado mais amplo para sua industrialização, os seguintes pontos foram observados:

1. Desprezando o custo do desenvolvimento de software o custo do sistema com componentes atualmente disponíveis é mais econômico em termos de nº de perfuradoras/nº de terminais a partir de 10 terminais.

2. O equipamento está menos sujeito à obsolescência.

3. É possível usar outro "software" sempre que desejado mudando-se a utilidade do sistema, o que não ocorre com as perfuradoras.

4. O serviço oferecido ao usuário é, na pior das hipóteses igual ao das perfuradoras sendo, no entanto, na maioria dos aspectos muito melhor.

5. O cartão fica substituido por fitas cassette muito mais práticas para armazenamento e transporte além de serem reutilizáveis; cabe notar que um programa de 2.000 cartões pode caber numa fita cassette que, nos modêlos especiais custam o mesmo prêço que os 2.000 cartões.

6. O meio de transferência para o computador será uma linha direta podendo ser a ligação feita remotamente. Tal fato possibilita que centros de menor porte disponham apenas de um sistema PRETEXTO li

gado a centros de maior porte em consonância com a política de regionalização da CAPRE. (Nada impede que tais centros usem o minicomputador para outras finalidades sempre que desejarem).

Para utilização remota é necessário que o sistema possa manter no local uma ou mais unidades de saída do computador (impressoras) onde os programas processados sejam devolvidos aos usuários. Embora tal facilidade não esteja descrita nêste projeto, é uma das primeiras extensões planejadas.

7. O sistema pode perfeitamente ser utilizado também em instalações comerciais.

8. Os componentes a serem fabricados pela industria nacional obviamente poderão ser comercializados com outras finalidades.

Observamos finalmente que foi uma preocupação explícita do projeto tornar o sistema suficientemente flexível para que ele possa ser usado em qualquer universidade ou em qualquer instituição que necessite de um método flexível de entrada de grande quantidade de dados digitalizados. Naturalmente esperamos que, uma vez concluído, este projeto seja industrializado e comercializado por uma firma brasileira.

EQUIPAMENTO E CONFIGURAÇÃO

Nesta escolha do equipamento para a implementação do sistema - foram pesados os seguintes ítens:

1. Representação e assistência técnica no Brasil
2. Custo dos equipamentos

3. Adequação ao projeto proposto

4. Periféricos disponíveis

Optou-se pelo PDP 11/10, fabricado pela Digital Equipment Corporation por dispor de todos os periféricos necessários, ter capacidade para endereçar bytes e arquitetura adequada ao projeto, além de respeitar os demais quesitos.

A configuração sugerida para o funcionamento do sistema é a seguinte:

1 processador com 16 K palavras de 16 bits (DIGITAL)

1 unidade de disco com capacidade para 2.4 byte (DIGITAL)

4 unidades de fita cassette (UFRJ)

32 (máximo) terminais teleimpressores (UFRJ)

Cumpre observar que a razão definitiva entre o nº de unidades fita cassette/nº de terminais teleimpressores será determinada pelo uso. Na realidade o sistema foi concebido de tal forma que esta razão pode ser facilmente alterada.

OBSERVAÇÃO: As unidades de fita e terminais para o sistema definitivo estão sendo desenvolvidos na UFRJ.

INTRODUÇÃO AO SISTEMA

Fazemos a seguir uma breve explanação do funcionamento do sistema, sem entrar nos detalhes técnicos, com o objetivo de dar uma visão de conjunto que é necessária para o melhor entendimento das especificações técnicas contidas nos capítulos seguintes.

O usuário deverá normalmente possuir pelo menos uma fita cassette para guardar os seus programas. Para "perfurar" o seu programa, o usuário se dirige a qualquer um dos terminais disponíveis e, após dar comandos apropriados, começa a teclar o seu programa.

Tal utilização é idêntica à de um editor de textos on-line. Cada linha de código é, automàticamente, numerada pelo sistema e, assim que o usuário acaba de teclar um comando o mesmo é sujeito a uma crítica de sintaxe (se assim fôr especificado) e se fôr o caso será impressa uma mensagem de êrro permitindo que o usuário conserte imediatamente o programa.

Uma vez pronto o programa, o usuário poderá especificar, não exclusivamente, um dos seguintes destinos para o mesmo:

a. o programa deve ser enviado para o computador central. Neste caso o texto que acaba de ser "perfurado" vai para o computador central para compilação e execução. O usuário poderá buscar na impressora do computador central o resultado do trabalho. (O sistema tem uma previsão para colocar, automaticamente, os comandos do sistema operacional no caso de programas dentro de um certo padrão; se o programa do usuário não se encaixar no padrão, o usuário deverá preparar os comandos do sistema juntamente com o programa).

b. o programa deve permanecer no disco do minicomputador. Neste caso o sistema avisa ao usuário até que horas o programa que acaba de ser "perfurado" permanecerá no disco. A finalidade deste sistema de reserva é evitar que o usuário tenha sempre que reintroduzir o programa no sistema se ele estiver trabalhando em regime intensivo de depuração (executando várias vezes o programa com pequenos intervalos de tempo).

O sistema guarda o programa por um tempo determinado pela instalação (sugerido entre 1/2 hora e 2 horas dependendo da carga do disco) com o objetivo de facilitar a depuração de programas quando a instalação central puder manter um baixo tempo de resposta. Neste caso o usuário manda executar o programa, pede uma RESERVA do mesmo e se despede do terminal. Recebendo sua listagem o usuário volta a qualquer terminal, faz os acêrtos necessários e completa o ciclo.

c. o programa deve ser gravado em uma fita cassette. Neste caso o usuário fornece ao sistema o número da fita em que o programa deve ser gravado, após o que o sistema aguarda que a fita apareça em uma das unidades para proceder a gravação. Uma vez efetuada a gravação, o usuário poderá a qualquer instante, posteriormente, colocar o texto fonte do seu programa no minicomputador para possíveis alterações, através da leitura da fita cassette. Para proceder a uma leitura o usuário se dirige a qualquer - terminal e indica o número da fita e o nome do arquivo que deve ser lido, após o que o sistema passa a aguardar o aparecimento da fita em uma das unidades de fita do minicomputador.

PREMISSAS

Algumas premissas foram estabelecidas logo no início da definição do projeto. Dentre elas devemos destacar as seguintes:

1. Os terminais foram considerados um recurso crítico quanto à sua ocupação; isto é: mesmo havendo muitos terminais poderá haver fila de usuários. Para reduzir ou eliminar êste problema o sistema é orientado de modo a manter o usuário no terminal o mínimo de tempo necessário; tentando

afastar o usuário sempre que se esperar ociosidade do terminal.

2. O disco é considerado um recurso crítico quanto à velocidade de atendimento. A capacidade foi julgada conveniente para a aplicação.

3. A memória é outro recurso crítico do sistema.

4. A relação nº de unidades cassette/nº de terminais (1 para 8) também torna tais unidades um recurso crítico e, portanto, é desejável que os usuários as ocupem o mínimo possível.

5. A responsabilidade sôbre as fitas cassette deve ser inteiramente do usuário e deve ser tratada como sendo de sua propriedade. Tôdas as fitas a serem usadas no sistema devem, no entanto, ser inicializadas e numeradas pela instalação.

6. Objetivando concluir o projeto o mais rápida e funcionalmente possível, as soluções simples (desde que razoáveis) foram preferidas.

7. O sistema deve, em termos absolutos, ser mais barato do que perfuradoras, e, na pior hipótese, prestar um serviço equivalente.

DEFINIÇÕES INICIAIS

Parte em respeito às premissas estabelecidas e parte por outras razões foram logo de início estabelecidas definições como as que se seguem:

1. Os dados gravados em cassette estarão em formato idêntico ao do disco.

2. Será feita compactação de brancos iniciais e finais nas linhas de texto porque pode-se ter uma economia significativa de armazenamento e é de simples implementação (por exemplo comandos do tipo A = 1).

3. Todos os nomes de arquivos são formados por 1 a 6 caracteres alfanuméricos sendo o 1º alfabético. (6 caracteres podem ser compactados em 2 palavras).

4. Vários programas ou arquivos podem ser armazenados em uma mesma fita. A responsabilidade de edição da fita é do sistema e não do usuário.

5. Os arquivos em cassette serão inicialmente alocados com 20% de espaço a mais que o necessário permitindo seu crescimento sem editar a fita.

6. Os usuários não reterão os terminais para ler ou gravar cassettes. As leituras são feitas antes do usuário entrar no sistema e as gravações após o mesmo ter saído.

7. O reconhecimento das fitas será feito pelo número gravado no rótulo, podendo as mesmas serem montadas em qualquer unidade disponível.

II. COMANDOS DO SISTEMA

Êste capítulo consta principalmente da definição de um editor de textos adequado às finalidades propostas. Os comandos foram escolhidos visando facilitar o serviço do usuário, porém sem sobrecarregar o minicomputador (desta forma, por exemplo, não há comando para procurar contexto a não ser em uma linha determinada).

As linhas do texto são identificadas por sua numeração que só pode ser mudada através de um comando do usuário.

Prevendo ainda a necessidade de eventualmente se proceder a manutenção ou mesmo exercer supervisão sôbre o uso do sistema foram tambem propostos comandos privativos do operador.

É conveniente notar que as informações referentes a cada usuário encontram-se na Tabela de Informações de Terminais (TIT) descrita na seção Tabelas do Sistema.

PROCEDIMENTOS

- O sistema pressupõe a existência de um terminal de operação porém não depende de ações ou presença de operador para seu funcionamento.

  Motivo: Comandos de contrôle poderão ser aceitos apenas dêste terminal. É indesejável pressupor a presença de operador porque caso contrário qualquer ausência ou incompetência do operador poderia prejudicar o funcionamento do sistema.

- O usuário que não necessitar de alimentar cassette para trabalhar se dirige diretamente a um terminal e começa a usar o sistema. O usuário que tenha dados a alimentar se dirige a um terminal desocupado, comanda carregar, informa o número da fita (rótulo), senha e nome dos arquivos a serem lidos, após o que deverá montar a fita em qualquer das unidades. Qualquer êrro detetado gera uma mensagem e o comando é cancelado.

- A senha (1 a 6 caracteres alfanuméricos) se destina apenas a relacionar os arquivos lidos da fita com o usuário quando o mesmo ocupar um terminal. Desta forma o usuário poderá usar uma senha diferente para cada sessão, se desejar.

COMANDOS DO SISTEMA

- OI - Para inicializar o sistema o usuário pressiona a tecla de retôrno do carro e o sistema responde com uma exclamação (!);

  O usuário deve entrar com:

  OI < senha >

  (Caso o usuário tenha entrado dados por cassette deverá usar a mesma senha da leitura).

  O sistema confirmará ou não a existência de arquivos do usuário da seguinte forma:

  NN ARQUIVOS DISPONÍVEIS

  Após o que o usuário poderá entrar com qualquer comando do sistema.

- INFORME - O comando informe (display) destina-se a prestar determinadas informações aos usuários especificadas como parâmetros; tais como:

  - ARQUIVOS - nome dos arquivos disponíveis.
  - HORA
  - ESTATÍSTICA - dados sôbre esta utilização. Ex.: tempo de uso, nº de linhas, etc.
  - DIMENSÃO < nome-de-arquivo > - número de linhas e blocos em determinado arquivo.
  - LAYOUT < número-de-fita > - informações contidas no diretório de um cassette. (O sistema armazena os rótulos de cada cassette lido.
  - EXECUÇÃO - informa as execuções pendentes.
  - DESCARREGAMENTO - informa os descarregamentos (em cassette) pendentes.
  - RESERVA - informa os arquivos em reserva.

OBS.: O informe execução, descarregamento e reserva tem, dentre outras, a finalidade de facilitar o cancelamento de execuções, cancelamentos e reservas.

- EDITAR - O comando editar coloca determinado arquivo disponível para edição; forma geral:

$$\text{EDITAR} < \text{nome-de-arquivo} > \begin{bmatrix} \text{FORTRAN} \\ \text{DADOS} \\ \text{TEXTO} \end{bmatrix} < \text{parâmetros} >$$

Se existir um arquivo com o nome designado êste será utilizado; caso contrário será criado um arquivo nôvo com êste nome. Conforme o caso o sistema informará:

- ARQUIVO JÁ EXISTENTE

ou

- ARQUIVO RECEM-CRIADO

Todos os comandos de edição subsequentes serão relacionados ao arquivo especificado no comando EDITAR até que outro comando EDITAR especifique nôvo arquivo. Um comando de edição sem que um comando EDITAR tenha sido dado causa êrro do tipo:

- ESPECIFIQUE ARQUIVO EM COMANDO EDITAR

As palavras FORTRAN, DADOS, TEXTO etc. determinam o nome do programa de crítica que será utilizado e poderão também ser ligados/desligados por um comando CRITICAR (SEGUE).

Após êste comando o sistema entra em modo de aceitação de linhas. O número da próxima linha disponível é impresso e o sistema aguarda o texto da linha. Para sair desta modalidade o usuário deve dar um retôrno de carro.

O nome da rotina de crítica poderá ou não implicar na inserção automática - de cartões contrôle. Isto ocorrerá sòmente com um arquivo nôvo; os cartões contrôle serão copiados para o início do arquivo e o campo de parâmetros, se existir, será inserido no 1º cartão contrôle. (parâmetro de contabilidade do JOB).

- CRITICAR - (SEGUE) êste comando permite ao usuário escolher o programa de crítica que será usado a seguir. O programa de crítica especificado é usado em substituição a qualquer outro que esteja porventura em uso (especificado no EDITAR, por exemplo), forma geral:

```
                 FORTRAN
                 DADOS
                 ASSEMBLER
                 TEXTO
CRITICAR         -
                 -
                 -
                 -
```

Ao fim dêste comando o sistema retorna automaticamente ao modo de aceitação imprimindo o número da próxima linha disponível.

Deve-se prever a possibilidade de passar parâmetros para a rotina de crítica.

Dentre as finalidades podemos lembrar a tabulação automática

Tabulação:

A tecla de tabulação poderá ser usada pelo programa de crítica para fazer uma tabulação automática após a entrada de uma linha completa;- ou seja após o acionamento de retôrno do carro. A rotina de conferência poderá fazer automaticamente o início da linha de continuação coerente com o tipo de texto (FORTRAN, ASSEMBLER, etc.) no caso de, após a tabulação haver necessidade de truncar a linha.

Uma variação do comando criticar é o comando.

SEGUE DADOS

Cuja ação é a mesma de um criticar dados com a diferença que são automaticamente inseridos no texto comandos separadores de dados (por exemplo /*, //SYSIN DD *)

- TCHAU - O comando tchau (LOGOFF) é utilizado pelo usuário para liberar o terminal. Após êste comando os arquivos desnecessários (não referenciados em EXECUTAR e DESCARREGAR são liberados para execução.

Motivo: Tôda a operação de transferência (EXECUTAR, DESCARREGAR) só pode ser iniciada após a liberação do terminal por 2 razões:

1. impede-se que o usuário queira editar arquivos que serão transmitidos.

2. assume-se que o usuário só deva levantar do terminal para colocar o cassette no fim da sessão.

O comando TCHAU pode ser forçado pelo sistema se um terminal permanecer - inativo por mais de 1 minuto e não responder à indagação feita pelo sistema. Caso o terminal esteja "NOT READY" na indagação, o usuário passará a ser encarado como se acabasse de ler seus arquivos de cassette e ainda não se tivesse identificado.

Motivo: Previsto para defeitos em terminal. Nêste caso basta o usuário se identificar em outro terminal e continuar seu trabalho.

Em resposta a um comando TCHAU dado pelo usuário o sistema emitirá as seguintes mensagens:

RESERVA NO. xxx VALIDA ATÉ AS xx.xx HS

ou

RESERVA NÃO EFETUADA

indicando o estado das reservas solicitadas; se a reserva for confirmada o número da reserva (xxx) deverá ser usado pelo usuário para recuperar os arquivos em um comando carregar. A reserva será cancelada às xx.xx horas ou logo após ser referenciada por um comando CARREGAR

MONTE FITAS: xxxxx . xxxxx ..... ***

ou

DESCARREGAMENTO IMPOSSÍVEL (LIMBO CHEIO)

indicando quais as fitas cassettes que deverão ser montadas pelo usuário para descarregamento, limpeza etc. A lista de fitas termina por 3 asteriscos. Se não houver fitas a montar apenas os 3 asteriscos serão impressos. Descarregamento impossível indica que o sistema não conseguiu providenciar os descarregamentos.

EXECUÇÕES PROVIDENCIADAS: < arquivo 1 > < arquivo 2 > .....***

indicando o nome dos arquivos cuja execução foi providenciada. A lista é encerrada por uma série de 3 asteriscos.

PODEMOS ENCERRAR ?

Se o usuário responder NÃO o sistema volta ao estado de comando; qualquer outra resposta o sistema passa ao modo inativo.

- RESERVAR < nome-de-arquivo > - Usado pelo usuário para garantir o armazenamento dos arquivos especificados para continuar seu trabalho posteriormente.

  Motivo: Evitar que o usuário que pretende voltar brevemente precise descarregar e carregar cassette.

O sistema confirmará (ou não) a reserva dos arquivos e a hora em que os mesmos serão liberados (tempo de permanência é especificado em geração).

Motivo: a garantia deve ser real; se o disco estiver sobrecarregado não serão aceitas novas reservas porém o usuário que obtiver confirmação poderá contar com a mesma.

A confirmação da reserva será feita após o TCHAU do usuário. (Ver comando TCHAU) e sua recuperação através de um comando CARREGAR.

- EXECUTAR [(R)] [< nome-de-arquivo >]

a) sem parâmetros:

O arquivo de trabalho atual é transmitido com cartões contrôle padronizados para o sistema central. Informações necessárias à contabilidade do sistema central devem ser dadas entre parênteses.

b) com parâmetros:

O arquivo especificado (ou a concatenação dos arquivos especificados) é transmitido para o sistema central sem nenhuma inserção de cartões-contrôle.

- CARREGAR < senha >

Os arquivos relacionados serão carregados no disco a partir do cassette especificado. Este comando só é aceito com o terminal não ativo.
Em resposta ao comando carregar o sistema poderá emitir uma das seguintes - mensagens:

- CARREGAMENTO IMPOSSÍVEL (LIMBO CHEIO)

indicando que o sistema não tem mais possibilidade de providenciar carregamentos.

- SENHA EM USO

indicando que outro usuário já está se utilizando desta senha e que, portanto, deve ser usada outra senha.

- DE NUMERO DE FITA E ARQUIVOS:

ao que o usuário deve responder com o número da fita e nomes dos arquivos a carregar separados por um espaço em branco. A cada retôrno de carro o usuário receberá novamente a mesma mensagem de modo a permitir o carregamento de várias fitas. Se o usuário responder simplesmente com um retôrno de carro, o comando será encerrado.

A recuperação de reservas efetuadas é respondida da seguinte forma: em resposta ao 1º DE NUMERO DE FITA E ARQUIVOS: O usuário entra com Rxxx onde xxx é o número da reserva. (Não precisa informar os arquivos). Após esta referência a reserva é cancelada.

- DESCARREGAR [(R)] <nº de cassette> <nome de arquivo>

O arquivo é descarregado para cassette, se for desejado concatenar vários arquivos em um só basta usar os nomes dos arquivos separados por símbolos '+'. O parâmetro (R) só é válido quando não houver concatenação e caso esteja presente o arquivo será renumerado. Quando há concatenação sempre

há renumeração. Vários arquivos podem ser especificados em um mesmo comando. Cada nome de arquivo é separado dos demais por um branco. Um arquivo concatenado é especificado da seguinte forma:

< nôvo nome > = < arquivo 1 > + < arquivo 2 > + ...

- COPIAR < nome-de-arquivo > [<número-de-linha> <número-de-linha>]

Especifica que o conjunto de linhas determinado (ou a totalidade do arquivo especificado) deve ser copiado para o fim do arquivo em edição. As novas linhas são renumeradas de tal forma que o seu nôvo numero seja igual à soma do número anterior com o número da última linha ocupada no texto. (a parte nova não conterá números quebrados, a parte antiga poderá ter).

- RENUMERAR [<início> <incremento>]

O texto em edição é renumerado <início> especifica o número que será dado à primeira linha. O texto será integralmente renumerado.

OBS.: o maior numero de linha é 60.000.

- ALTERAR <numero de linha> Δ <texto 1> Δ <texto 2> Δ

A linha especificada é varrida substituindo-se tôdas as incidências de < texto 1 > por < texto 2 > . Δ é qualquer símbolo diferente de branco não contido em <texto 1 > nem em < texto 2 >.

Após a substituição a linha é impressa para verificação. Caracteres que excederem as posições disponíveis são truncados.

Após a alteração a linha é sujeita à crítica de sintaxe.

- APAGAR { TUDO / <numero-de-linha> [<numero-de-linha>] }

A linha ou o conjunto de linhas especificadas são apagadas do texto em edição. Não sendo especificado nº de linha (TUDO) o texto será apagado integralmente.

<NUMERO-DE-LINHA> <TEXTO>

Permite ao usuário entrar uma linha de texto fora da modalidade de edição. < nº de linha > é, no máximo, igual ao maior nº linha existente.

- LISTAR [<número-de-linha>] [<número-de-linha>] { NUMEROS / TEXTO }

O conjunto de linhas ou a totalidade do texto especificado é listado. Na opção NUMEROS os números de linhas são também impressos. Na opção texto apenas o texto é impresso.

- CANCELAR { EXECUTAR / RESERVE } < nome-de-arquivo >

A execução ou reserva do arquivo especificado é cancelada.

- CANCELAR { DESCARREGAR / LIMPAR } « nº da fita » « nome de arquivo >

O descarregamento (ou limpeza) do(s) arquivo(s) especificados é (são) cancelado(s). Se não houver nome de arquivo(s), todo o descarregamento (limpeza) para a fita é cancelado.

Neste caso, o comando é válido para apenas uma fita de cada vez.

- LIMPAR <Nº FITA> <ARQUIVOS>

Será marcado um comando pendente para a fita e, os arquivos referenciados pelo comando serão destruídos no momento da gravação da fita.

O comando é válido para apenas uma fita de cada vez.

Identificação do operador

Para que um terminal aceite comandos de operação é necessário que o operador se identifique com uma senha especial.

Motivo: Se um terminal fôsse permanentemente alocado ao operador isto poderia ser um ônus pesado para sistemas com poucos terminais; por outro lado se este terminal não estivesse bem guardado os usuários poderiam fazer uso de comandos especiais, o que seria indesejável.

Inicialmente pensou-se em um terminal de operação que seria, inclusive usado pelos usuários para seus comandos CARREGUE. O sistema, desta forma não saberia se um comando estava sendo dado por um usuário ou pelo operador.

O operador se identifica com:

OI OPERADOR < senha >

reconhecendo a senha o terminal passará a aceitar dêste terminal todos - os comandos de operação.

Observe-se que apenas 1 terminal a cada tempo pode estar com o operador.

Comandos

Todos os comandos comuns são aceitos normalmente.

Os comandos Informe e Cancele passam a ter novos parâmetros e novos comandos são disponíveis conforme descrição abaixo. (Tais comandos são válidos mesmo com o terminal inativo).

Comando Informe:

INFORME

TERMINAIS - lista a tabela de terminais

CASSETTES - lista a tabela de cassettes

LIMBO - lista a tabela do LIMBO

DISCO - lista o mapa do disco

<senha> - lista a entrada referente a esta senha

<número> - lista a entrada referente ao equipamento cujo nº foi dado.

Comando Cancele:

CANCELE <senha> TUDO - cancela todas as entradas, em tôdas as tabelas referentes a esta senha.

Comando Seja:

SEJA HORA = xx,xx - usado para alterar a hora

SEJA SENHA = <senha> - usado para mudar a senha do operador

SEJA TERMINAL = <nº> - usado para mudar o terminal de saída

SEJA ARQUIVO <nome-de-arquivo 1 >= <nome-de-arquivo 2> - usado para mudar nomes de arquivos. Se ambos os nomes existirem os nomes serão trocados nos arquivos.

Se apenas < nome-de-arquivo 1 > existir o nome do arquivo será mudado para < nome-de-arquivo 2 > .

Observações: Arquivo público do sistema começarão com $ (inválido para usuários). Tais arquivos são protegidos contra os seguintes comandos:

- < nº de linha > (implícito)

- ALTERAR

- EDITAR (o comando é válido; no entanto o sistema ao invés de responder ARQUIVO EXISTENTE ou recém criado e bater o nº de linha responde ARQUIVO PROTEGIDO; CONSULTA PERMITIDA e não dá nº de linha (nunca entra em aceitação).

- APAGAR

Tais comandos são protegidos mesmo do terminal do operador. O operador, no entanto tem o recurso de mudar o nome dos arquivos através de um comando seja. É conveniente, no entanto que os arquivos do sistema sejam inexistentes pelo menor tempo possível; assim o operador quando precisar alterar um dêstes arquivos deverá antes copiá-lo, alterá-lo e substituir os nomes. Uma solução razoável é que em um comando SEJA, caso os dois nomes de arquivos sejam existentes os nomes sejam trocados (ou seja: vice-versa).

Interferência do Operador através dos comandos levar e trazer:

O comando OI OPERADOR simplesmente identifica o terminal como sendo de operação. O terminal pode continuar inativo. Se desejar o operador pode entrar com um OI ativando o terminal. Os comandos levar e trazer permitem que o terminal do operador seja virtualmente trocado com qualquer outro. Dentre as aplicações previstas temos:

1 - O operador pode ativar o terminal para um usuário.

2 - O operador pode trocar um usuário de terminal.

Para tanto estes comandos são válidos mesmo que o terminal esteja inativo. Para desligar o operador deve-se dar um TCHAU OPERADOR.

LEVAR (nº de terminal)

O estado atual do terminal do operador é levado para o terminal especificado (A entrada na TIT é levada). A entrada na TIT do operador é desativada no sentido que um novo comando levar torna o terminal - de destino desativado. O terminal de operação continua aceitando seus comandos normalmente.

TRAZER (nº de terminal)

A entrada na TIT do terminal especificado é trazida para a entrada do terminal do operador. O terminal permanece trancado enquanto o operador não der um comando levar para este terminal.

Observação: Os comandos levar e trazer são válidos tanto no modo ativo como no inativo.

COMANDO MEMÓRIA:

O comando memória serve para consultar ou alterar uma posição de memória.

Sua forma é a seguinte:

$$\text{MEMÓRIA} \quad (\text{endereço}) = \left\{ \begin{array}{c} ? \\ (\text{conteúdo}) \end{array} \right\}$$

Onde endereço e conteúdo são números em octal.

Se for dada interrogação o sistema imprimirá, em octal o conteúdo da palavra desejada. Se for dado um conteúdo, este será colocado na posição especificada.

III. TABELAS DO SISTEMA

Nêste capítulo são descritas as tabelas do sistema. A concentração dos dados nestas tabelas objetiva um melhor aproveitamento dos recursos de programação reentrante oferecidos pelo PDP-11 pois as informações que são variáveis para cada terminal ou unidade cassette (também denominados conjuntamente como "usuários") estão armazenadas nestas tabelas. Além disso, se obtém maior modularidade na programação.

TABELA DE INFORMAÇÃO DE TERMINAIS (TIT)

Cada entrada nesta tabela está associada a um terminal contendo ainda informações relativas ao processamento e usuário que ocupa o mesmo. Também poderão ser usadas entradas para outras unidades (cassettes) além dos terminais. O objetivo desta tabela é concentrar tôdas as informações associadas a cada terminal.

Descrição das informações: (O nº do terminal é dado pela ordem na tabela).

6b - Estado do terminal - especifica a função que o terminal está executando (listagem, entrada, etc.) ou ainda se o mesmo está ocioso, desligado, off-line etc.

8b - Apontador da linha em processamento referenciada - aponta para o byte no buffer onde se inicia a linha. (Contém zêro se não houve referência).

4b - Indicador de rotina de sintaxe - indica qual a rotina que deverá ser chamada quando uma linha de dados estiver completa.

16b - Hora da última interrupção - contém a hora em que ocorreu a última interrupção dêste terminal. (Hora de inicialização do evento)

Motivo: permitir o TCHAU forçado.

16b - Apontador para o próximo caráter. Contém o endereço onde será colocado o próximo carater a ser entrado por êste terminal.

Motivo: facilitar a rotina de entrada por terminais.

32b - Senha do Usuário

10b - Apontador p/bloco do diretório

16b - Número da última linha referenciada (ULR)

8b - Ponteiro para o próximo da lista

16b - Endereço do programão

16b - Endereço do setor de disco em uso

16b - Apontador p/mapa do disco, arquivo atual (aponta para o último bloco referenciado).

16b - U.L.P. - (última linha a processar) - contém o número da ultima linha a processar quando se executa uma operação repetitiva. Usado para trabalho em outras operações.

1b - Iminência de TCHAU forçado = 1 se o usuário já recebeu um aviso para impedir o TCHAU.

1b - Proteção do arquivo = 1 se o arquivo em edição é protegido;
= 0 caso contrário

TABELA DE DISCO

Esta tabela destina-se a controlar a alocação de espaço em disco. Cada entrada corresponde a 1 bloco e contém as seguintes informações:

16b - LINHA LIMITE - contém a numeração absoluta da linha limite deste bloco.

Motivo: a existência da numeração da linha limite é necessária para dar rapidez a acessos aleatórios.

6b - MAPA DO BLOCO - contém 1 bit para cada setor do bloco.
O bit em 1 indica que o referido setor está ocupado (contém dados válidos), o bit em 0 indica setor vazio.

Motivo: permitir contrôle rápido sôbre o estado de cada setor no bloco.

10b - Apontador do próximo bloco aponta, dentro desta mesma tabela para o bloco seguinte do mesmo usuário. O último bloco aponta para o 1º.

Observações: Blocos não-alocados terão a 2a. palavra (mapa + apontador) - igual a zero. Assim, para desalocar um bloco basta zerar esta palavra na sua entrada.

LIMBO

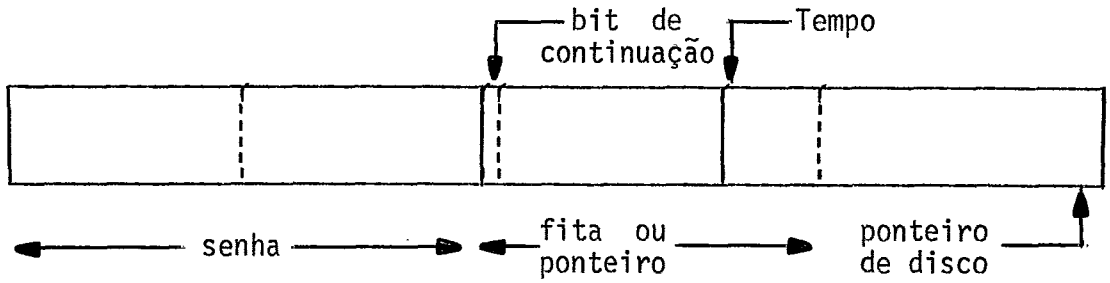
O limbo é uma tabela que contém informações relativas a usuários que não estão em atividade porém estão para entrar ou sair do sistema.

Sugere-se que a tabela do limbo tenha cêrca de 64 entradas e a tabela de fitas 16 entradas.

Observações:

- não se deve permitir que o limbo se encha com pessoal desejando entrar no sistema para evitar um trancamento do mesmo.

Entrada do limbo:

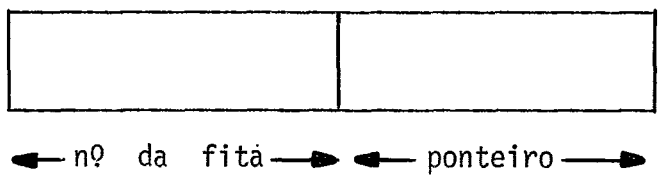


se bit de continuação = 1

o campo fita ou ponteiro contém um apontador para entradas do tipo abaixo. (lista de fitas).

= 0 contém o nº de fita cassette.

Lista de fitas



Processamento do limbo:

- as entradas são inicializadas após entrada de um comando CARREGUE no terminal.

1. MONTAGEM DE CASSETTE

Quando uma fita é montada, seu label é lido e as entradas no limbo são percorridas buscando um MATCH de nº de fita. Se não houver MATCH deve ser impressa uma mensagem no terminal de operação: Ex.:

FITA NNNN UNIDADE N NÃO TEM TAREFA

Havendo MATCH, através do ponteiro de disco localiza-se o diretório do usuário bem como os comandos do mesmo. Inicializa-se o sistema para leitura do 1º arquivo pedido na fita atualizando-se a Tabela de informação de cassettes.

- Completando-se a transferência de um arquivo; (satisfatoriamente) novamente se faz referência aos comandos do usuário. Retira-se o comando já cumprido; se ainda houver comandos para a mesma fita nova inicialização deve ser feita; caso contrário procede-se da seguinte forma:

a. Se o bit continuação estiver desligado fazer o nº da fita = 0. Usuário está pronto para entrar no sistema

b. Se o bit de continuação fôr 1, retirar a entrada correspondente ao nº da fita processada tendo o cuidado de verificar se vai faltar apenas - uma entrada (a nova lista de fitas só contém 1 entrada) pois, neste caso a lista de fitas pode ser eliminada colocando-se o nº da fita na própria entrada do limbo e desligando-se o bit de continuação.

- Usuário dá um comando ÔI.

A senha do usuário deve ser encontrada no limbo (criticar antes se não está ativa) caso contrário o usuário não possui arquivos disponíveis.

Localizando-se a senha deve-se inicializar a tabela de Informações de Terminais (TIT) e retirar o usuário do limbo. Deve-se observar que, neste caso o número da fita deve ser zero; pois, caso contrário ainda existe fita para ser lida e, portanto o usuário não está apto a entrar no sistema.

TABELA DE INFORMAÇÃO DE CASSETTES:

4b - Estado

12b - Apontador última linha referenciada (ULR)

16b - Hora da última interrupção

12b - Apontador p/o próximo carater

10b - Apontador p/o bloco atual

10b - Apontador p/bloco do diretório

16b - Nº de blocos gravados

16b - Posição do arquivo atual

8b - Ponteiro para o próximo da lista

16b - Endereço do programão

16b - Endereço do setor de disco em uso

16b - U.L.P. (última linha a processar) - contém o número da última linha a processar quando se executa uma operação repetitiva. Usado para trabalho em outras operações.

8b - Comprimento do setor (em palavras)

TABELA DE RESERVAS

A tabela de reservas se destina a manter informações sobre as reservas aceitas pelo sistema. Cada entrada nesta tabela tem a seguinte forma:

Nº DA RESERVA	TEMPO	APONTADOR

onde:

Nº da reserva (16 bits): contém um nº binário dado pelo sistema que corresponde ao nº da reserva.

Tempo (3 bits): contém o nº de unidades de tempo que ainda faltam transcorrer antes de se cancelar a reserva. Este valor é calculado no início da reserva e documentado por ação da rotina de Timer ao fim de cada unidade de tempo. Quando chegar a zero, a reserva será cancelada.

Apontador (13 bits): é o apontador absoluto para o disco; setor onde se encontra o diretório do usuário que solicitou a reserva. Caso a reserva seja cancelada, este apontador contém zeros.

Total da tabela: 100 palavras (50 entradas de 2 palavras)

IV. ORGANIZAÇÃO DO DISCO

Todos os arquivos de trabalho encontram-se no disco que foi considerado, desde o início do projeto um recurso crítico do sistema quanto a velocidade de acesso (não quanto à capacidade).

O capítulo a seguir descreve a organização adotada no disco que foi definida, principalmente, objetivando-se um atendimento rápido. Cabe ressaltar que a demanda é normalmente aleatória pois, com 32 usuários no sistema, mesmo que cada um solicitasse acesso sequencial, o disco raramente atenderia a dois pedidos consecutivos que fôssem do mesmo usuário.

A organização proposta efetua o acesso com, no máximo, 1 movimentação de disco.

Também está nêste capítulo a definição dos "buffers" dos usuários na qual foi adotada a solução mais simples após a discussão de diversas soluções mais elaboradas.

CARACTERÍSTICAS TÉCNICAS

200 cilindros

1 cilindro = 2 trilhas

1 trilha = 12 setores

1 setor = 256 palavras

tempo de latência média = 20 ms

movimentação de 1 trilha = 10 ms

posicionamento médio = 70 ms (incluíndo latência)

ORGANIZAÇÃO LÓGICA

A unidade de endereçamento é o setor de 256 palavras. A organização lógica usada no sistema se baseia em uma unidade chamada bloco cuja definição se segue:

- um bloco consta de 6 setores alternados de uma trilha. O bloco que contém os setores pares é o bloco "0" e o que contém os setores impares é o bloco "1" da trilha.

  Motivos: Os setores alternados visam permitir maior tempo de processamento entre o fim da leitura de um setor e o início da leitura de outro. Mais explicitamente, logo após a leitura de um setor pode-se decidir que o próximo setor deve ser lido porém seria possível que o início do mesmo já tivesse passado pela cabeça de leitura resultando em se esperar uma revolução.

  O tamanho do bloco (1.5 K palavras ou 3 K bytes) está bem balanceado com o compromisso entre o tamanho do mapa do disco (melhor poucos blocos grandes) frequência de overflow e pedidos de alocação versus desperdício de áreas pouco usadas em disco, número de áreas, facilidade de corrigir estouro etc. (melhor muitos blocos pequenos).

  Assim, o disco fica dividido em 800 blocos de 3 K bytes cada um. O bloco é a menor unidade de espaço em disco alocável a um arquivo qualquer. Isto é, os arquivos crescem 1 bloco de cada vez.

As linhas de um arquivo são armazenadas pelos setores e blocos do mesmo observando-se as seguintes restrições:

a) Cada setor contém um intervalo contínuo de linhas. (se as linhas 315 e 380 estão em um setor, a linha 350, se existir estará necessariamente no mesmo setor).

Motivo: facilidade de localização

b) Cada bloco contém um intervalo contínuo de linhas (ídem)

Motivo: facilidade de localização

c) A ordem dos setores em um bloco independe da ordem da numeração das linhas contidas nos mesmos. (o setor 0 pode conter linhas de nºs 800 a 850; setor 1 de 120 a 145; setor 2 de 1100 a 1328; setor 3 de 80 a 110; etc.)

Motivo: menor rigidez, resultando em menor necessidade de rearrumação e coerência com o critério de alocação descrito adiante.

d) Cada setor contém um número inteiro de linhas; não há overflow de linhas de um setor para outro (uma linha não pode começar em um setor e acabar em outro).

e) Quando da alocação inicial, um setor será considerado completo quanto tiver menos de 96 palavras disponíveis.

Motivo: permitir que um certo número de alterações possam ser feitas sem causar overflow.

f) Idealmente cada bloco terá 1 setor vazio quando da alocação inicial.

ACESSO A LINHAS DE TEXTO

Com a organização adotada e utilizando-se o mapa do disco o acesso a uma linha qualquer, de um texto em edição de qualquer usuário, pode ser feito com, no máximo 1 movimento de braço, ou seja o tempo médio de acesso a uma linha qualquer é o mesmo tempo médio do disco.

Procedimento:

Na tabela de informações (de Terminais ou Cassettes) tem-se um apontador para o último bloco referenciado do arquivo (no mapa do disco); No mapa do disco tem-se uma lista circular dos blocos dêste arquivo. Usando-se o apontador mencionado procura-se nesta lista geral o bloco que deve conter a linha em questão usando-se, para comparação a linha-limite de cada bloco.

Localizando-se o bloco providencia-se o movimento do braço para o cilindro em questão e os setores ocupados do bloco são pesquisados um a um. Note-se que a pesquisa dos setores não aumenta o tempo de acesso pois a pesquisa terá início no 1º bloco que aparecer, reduzindo-se portanto a latência.

ALOCAÇÃO DE BLOCOS E SETORES

Alocação de setores

A alocação de um setor dentro de um bloco é feita objetivando a maior rapidez possível.

Na entrada do mapa do disco correspondente ao bloco em questão verifica-se a existência de setores disponíveis e é alocado aquêle cujo início esteja mais próximo da cabeça de gravação. Um bit 1 será colocado na ocupação do setor.

Desalocação de setores

Basta colocar um bit "0" na ocupação do mesmo. Se todos os setores estiverem desocupados o bloco deve ser desalocado.

Alocação de blocos

Deve-se alocar o bloco disponível que esteja mais ao centro do disco.

Motivo: Inicialmente pensou-se fazer a alocação o mais próximo possível da posição atual do braço. Tal solução seria muito boa principalmente se todos os arquivos fôssem transientes. Ocorre que o arquivo de mensagens de êrro é permanente e muito utilizado. Assim surgiu a idéia de se alocar sempre perto dêste arquivo que, por ser muito utilizado ficaria no centro do disco

Desalocação de blocos

Os campos de ocupação de setores e apontador para o próximo bloco devem ser zerados no mapa do disco. Observar que a lista de blocos do arquivo deve - ser mantida.

Estouro (overflow) de blocos e setores

Situação para chamada de estouro:

- quando é detetado que as linhas alocadas em um setor excedem a capacidade do mesmo.

Objetivo do algorítmo:

- corrigir a situação de modo a que a mesma só volte a ocorrer o mais tarde possivel.

Estouro de setor:

Ocorrendo estouro de setor, êste será dividido em 2 setores. O procedimento encontra-se descrito adiante na seção "algoritmos de estouro".

Observação:

Continuidade na numeração das linhas

Foi constatada a conveniência de que, de um setor para outro não houvesse descontinuidade na numeração das linhas. Vejamos, por exemplo o seguinte caso:

Setor 1			
linhas:	10	20	30
Setor 2			
linhas:	50	60	70
.			
.			
.			
Setor 6			
linhas:	35	40	43

Para têrmos acesso em sequência às linhas dêste bloco, após as linhas 10 20 30, seremos incapazes de determinar se as linhas 50, 60 e 70 (setor 2) são

as próximas, a menos que consultemos todos os setores e constatemos que após o setor 1 se segue o setor 6. Note-se que, se não houvesse o setor 6, deveríamos ler os setores um a um e, sòmente após a leitura de todos êles poderíamos determinar que o setor 2 seria o próximo, gastando uma revolução extra para lê-lo novamente.

Problemas análogos surgiriam por exemplo para localizar linhas inexistentes (chegar a esta conclusão).

Para resolver êste problema temos a seguinte solução: cada setor ao invés de conter em seu início o número da 1a. e da última linha do mesmo, conterá a numeração das linhas que deveriam estar neste setor; no exemplo acima teríamos:

Setor 1	(0-30)	ou	
linhas:	10	20	30
Setor 2	(50-79)		
linhas:			
.			
.			
.			
Setor 6	(31-49)		

Dessa forma quando o setor 2 fôr lido podemos saber que êle <u>não</u> é o próximo na sequência pois como o setor 1 abrange linhas de 0 a 30 o próximo deve abranger a linha 31 tal como ocorre no setor 6.

Êsse procedimento também traz vantagens ao algorítmo de estouro de bloco proposto. A única desvantagem possível dêste esquema é que quando um setor fôr destruído, será necessário atualizar a numeração em um dos blocos vizinhos, ou desperdiçá-lo.

Estouro de Bloco

O estouro de bloco ocorre sempre devido a um estouro de setor. O algorítmo proposto adiante objetiva resolver o problema o mais ràpidamente possível.

Objetivando a rapidez, a idéia básica vem a ser a liberação de um setor de bloco em estouro transferindo seu conteúdo para um outro bloco. Para respeitar a continuidade da numeração apenas os setores das extremidades. (1º ou último) podem ser considerados para transferência.

Tenta-se em primeiro lugar resolver o problema sem alocar um nôvo bloco; isto é: transferindo o 1º setor para o bloco anterior ou o último para o bloco seguinte. Se ambos os blocos estiverem totalmente ocupados, então um nôvo bloco será alocado.

Dá-se preferência a ocupar o bloco anterior porque, uma vez que o usuário, normalmente, trabalha do início para o fim do texto é de se esperar menor risco de estouro no bloco anterior. Adiante o algorítmo é descrito para o caso de se dispor de 1 ou 2 buffers de disco.

Críticas ao algorítmo

Seria interessante que o algorítmo atendesse ainda aos seguintes objetivos secundários:

- Utilizasse eficientemente o disco
- Adiasse o mais possível um nôvo estouro

Quanto à utilização do disco aparentemente o algorítmo proposto é bom; Considerando não haver blocos apagados, (o que parece pouco significativo) pode-se, no pior dos casos ter um total de 20 setores ocupados em cada 5 blocos; ou seja uma taxa de ocupação de 2/3 que pode ser considerada eficiente para um "pior caso". O exemplo abaixo mostra a ocorrência do pior caso. Note-se que um setor que seja inserido em qualquer lugar melhora a taxa de ocupação.

Exemplo do pior caso:

1º bloco	(x)	(x)	(x)	(x)	(x)	(x)	6 setores ocupados
2º bloco	(x)	( )	( )	( )	( )	( )	1 setor ocupado
3º bloco	(x)	(x)	(x)	(x)	(x)	(x)	6 setores ocupados
4º bloco	(x)	(x)	(x)	(x)	(x)	(x)	6 setores ocupados
5º bloco	(x)	( )	( )	( )	( )	( )	1 setor ocupado

O adiamento da ocorrência de overflow não é plenamente atendido uma vez que o bloco em estouro, após a correção continua sem setores vazios. (O setor desalocado fica ocupado pelo nôvo setor de estouro). Estando, portanto sujeito à ocorrência de nôvo estouro pròximamente. No entanto, se assumirmos que só dispomos de um buffer de disco e que, para liberarmos um setor a mais precisaríamos de duas "viagens" ao nôvo bloco devemos acabar concordando ser preferível esperar a ocorrência de outro estouro pois, se o mesmo não ocorrer (o que é provável) estaremos economizando 1 movimento de ida e volta e, se ocorrer, o pior que temos a perder é o tempo de execução da rotina de estouro (sem acessos a disco).

Além do mais, o algorítmo dá preferência a deslocar o último setor do bloco para um bloco vazio porque novos estouros têm mais possibilidade de ocorrer

nos setores seguintes do que nos anteriores ao atual setor de estouro (trabalha-se do início para o fim).

Observe-se também que se o setor de estouro coincidir com o setor que se desejar mover (1º ou último) o algoritmo é ainda mais rápido.

Necessidade da continuidade das linhas

A continuidade das linhas é necessária para se determinar facilmente (com auxílio do mapa do disco) qual é o 1º ou o último setor de um bloco. Também é necessária para se atualizar a linha base. (Se movermos o 1º setor para o bloco anterior a nova linha base seria facilmente calculada como a proxima linha não contida nêste setor ao invés de precisarmos ler o nôvo setor para saber a linha base).

ALGORITMO DE ESTOURO

As seguintes alternativas foram analisadas:

1. usar 2 buffers de disco (256 palavras cada)
2. usar 1 buffer de 306 palavras (para acomodar o overflow) e um de 256.
3. usar 1 buffer de 256 palavras. Acomodar o overflow no buffer do usuário

A divisão do setor em estouro admite ainda, as seguintes alternativas principais:

1. Dividir pela metade
2. Dividir ajustando um dos novos setores na ocupação ótima
3. Dividir na U.L.R.

O modo de dividir o setor tem pouca interferência com os algorítmos descritos a seguir.

Para compactar o algoritmo vamos usar a seguinte notação:

lim (x) = linha limite do bloco X

PLS (A) = PLS do setor apontado por A

ULS (A) = ULS do setor apontado por A

SC (n) = setor S que contém a linha nº n

Algorítmo de estouro:

1 buffer de 256

1 buffer de 306 palavras.

1. Dividir o setor em estouro em 2 novos setores no buffer de 306 palavras. A aponta para o 1º e B para o 2º C aponta para o buffer de 256.

2. Havendo ainda 1 setor disponível nêste bloco, gravar os setores A e B nêste bloco; fim. Caso contrário continue (espera ≅ 1 latência)

3. (Estouro de bloco)

(Ver bloco anterior - pesquisar o 1º setor de bloco)

x = bloco atual

y = bloco anterior; se y estiver completamente cheio ou não existir, vá para 4.

z = PLS (A)

w = lim (Y) + 1

troque A e B.

vá para 5

4. (Ver bloco seguinte ou alocar nôvo bloco)

(pesquisar o último setor do bloco)

Z = ULS (B)

W = lim (x)

Y = bloco seguinte a X; se Y estiver completamente cheio ou não existir, aloque nôvo bloco; Y = nôvo bloco.

5. Se Z = W vá para 6;

ler SC(W) para C; gravar B; (espera ≅ 1 latência)

fazer B = C

6. Gravar A em X; gravar B em Y (espera = 2 latências + 1 movimento)

fim

Algoritmo de estouro usando 2 buffers de 256 palavras.

O algoritmo é o mesmo com as seguintes alterações:

1. Dividir o setor de estouro em 2 novos setores, 1 em cada buffer.

A aponta para o 1º e B para o 2º.

Se Z = W vá para 6; gravar B; (espera = 1 latência) ler SC (W) para B; (espera = 1 latência)

Vê-se que, no pior caso perde-se aproximadamente o tempo de 1 latência; em todos os outros casos tem-se o mesmo tempo.

Algorítmo usando 1 buffer:

1. A = início do buffer; dividir o setor em 2 e fazer

B = início do 2º setor

Se o bloco atual estiver cheio, vá para 3.

2. Regravar A e B. FIM

3. (ver bloco anterior)

   x = bloco atual

   y = bloco anterior; se estiver cheio ou não existir, vá para 4.

   z = PLS (A)

   w = lim (Y) + 1

   troque A e B

   vá para 5

4. (ver bloco seguinte)

   z = ULS (A)

   w = lim (X)

   y = bloco seguinte a x; se estiver cheio, ou não existir, aloque novo bloco;

   y = novo bloco.

5. Se z = w gravar A em X e B em Y; FIM;

6. Gravar o setor em estouro (256 palavras) em X;

   ler SC(W) e liberá-lo; gravar em Y;

   ler o setor em estouro em X;

   vá para 2

Comparação entre a eficiência dos algorítmos:

Basicamente os tempos de execução diferem para 3 situações:

1. existência de setor no bloco.
2. setor da extremidade desejada está no buffer
3. setor da extremidade não está no buffer.

Assim temos:

CASO

Algoritmo - (buffers)	1	2	3
256 + 306	20 ms	90 ms	110 ms
2 x 256	20 ms	90 ms	130 ms
1 só	20 ms	90 ms	220 ms

apenas os tempos de disco foram considerados; os seguintes valores foram usados:

latência p/1 ou 2 setores = 20 ms

movimentação média = 50 ms

O algorítmo a ser utilizado dependerá ainda de outras análises a serem feitas no desenvolvimento do projeto, estando o algorítmo de 256 + + 306 fortemente recomendado.

ARQUIVO DE CONTROLE

O arquivo de controle tem a finalidade de armazenar informações de contrôle que não necessitem ficar na memória.

Observação: Inicialmente pensou-se em armazenar as informações de controle juntamente com um arquivo qualquer de cada usuário ou em uma área fixa. A

área fixa foi abandonada por prejudicar o algorítmo de alocação de disco; resolveu-se usar um arquivo de controle ao invés de junto do usuário porque:

- não se misturam controle com dados.
- a desalocação de um arquivo não precisa consultar se o mesmo contem controles que devem ser mantidos.
- o esburacamento do disco é aparentemente menor.

Diferenças no tratamento:

O arquivo de controle tem, em alguns casos, tratamento diferente dos arquivos comuns:

1. Não existe número de linha; não há portanto uso da linha limite no mapa de disco.
2. A alocação de um setor é feita em qualquer bloco que tenha setores disponíveis e não em um bloco determinado pela numeração de linha.
3. Não há necessidade de manter setores vazios nos blocos uma vez que não há estouro.

Tipos de Registros:

Dois tipos de registros podem existir no arquivo contrôle:

- Diretório
- Registro de Rótulo (Label)

Cada um desses registros ocupa um setor do disco.

DIRETÓRIO

O diretório tem a finalidade de complementar as informações de contrôle de cada usuário. Uma vez que se assume êste o registro mais importante, para facilidade de acesso, todas as tabelas mantêm um apontador para o "bloco do diretório" onde um apontador para a tabela de disco indica o bloco no qual está contido o diretório do usuário em questão.

O diretório se divide em uma área de 16 palavras contendo contrôles e uma área de 240 palavras contendo três listas:

Lista de fitas, lista de arquivos e lista de concatenações como se segue:

CONTRÔLE	FITAS	ARQUIVOS	CONCATENAÇÕES

CONTRÔLE:

(2w) senha

(1w) Estado ... (ativo, limbo entrada, limbo saída, limbo forçado ou reserve)

(2w) Hora .... hora de entrada no estado atual

(8b) apontador p/início da lista de fitas

(8b) numero de entradas da lista de fitas
(será limitado em 10 para usuários e zero para o sistema)

(8b) apontador p/início da lista de arquivos

(8b) numero de entradas da lista de arquivos
(será limitado em 40 para usuários e 60 para o sistema)

(8b) apontador p/início da lista de concatenação

(8b) número de entradas da lista de concatenação

(será limitado em 100 para usuários e zero para o sistema)

(6w) Reservado

Lista de Fitas:

Cada entrada consta de 3 palavras contendo as seguintes informações:

(1w) Número da fita

(1w) Ponteiro do rótulo da fita (para o mapa do disco)

(1w) Contrôle

Lista de Arquivos

Cada entrada consta de 4 palavras contendo as seguintes informações:

(2w) Nome do arquivo

(1w) Ponteiro

- Se o bit de concatenação fôr zero, êste ponteiro aponta para o mapa do disco (1º bloco do arquivo)

- Se o bit de concatenação fôr "1", o 1º byte aponta para a posição dentro da lista de concatenação onde se inicia a sub-lista de concatenação que define o arquivo; o 2º byte indica o comprimento em bytes dessa sub-lista.

(1w) Contrôle que compreende:

(1b) - bit de execução - igual a "1" se tiver sido comandada execução

(1b) - bit de reserva - idem, reserva

(1b) - bit de concatenação - igual a "1" se o arquivo é formado por concatenação de outros.

(5b) - Reservado

(8b) - byte de Referências - conta o nº de vêzes em que o arquivo foi referenciado em concatenações, execuções ou descarregamentos.

Lista de Concatenação:

Esta lista é composta de várias sub-listas de tamanho variável. o início e o tamanho de cada lista são dados pelo ponteiro na lista de arquivos. Cada elemento de uma sub-lista ocupa 1 byte e contém as seguintes informações:

(1b) bit de diretório - indica se a referência é ao diretório do usuário (0) ou ao diretório do sistema (1); ou seja se o arquivo é do usuário ou do sistema.

(7b) ponteiro para a lista de arquivos

Aponta para a entrada da lista correspondente ao arquivo desejado.

O diretório do usuário é criado quando o usuário ingressa no sistema (através de CARREGAR ou OI) é destruído quando sua entrada no limbo é apagada (a não ser por um OI) durante uma reserva o diretório continua - existindo e é reativado quando fôr comandado CARREGAR. Durante a reserva os campos de lista de fitas e de concatenações não têm valor, e a senha tem código especial.

Observação: (MOTIVO)

A disposição da lista de concatenação é assim proposta visando economia de espaço e rapidez de processamento. Observe-se que a disposição do diretório do sistema não deve ser alterada quando seus arquivos forem atualizados; isto é, se o operador trocar um arquivo $A por B (SEJA $A=B) a entrada referente ao arquivo $A deve ser atualizada para apontar e referenciar o arquivo B permanecendo na mesma posição da lista de arquivos do diretório do sistema.

O byte de referência no diretório do usuário é incrementado a cada execução, descarregamento ou concatenação e decrementado a cada cancelamento ou término efetivo de uma referência feita por execução, cancelamento ou concatenação. Tal procedimento visa impedir que o arquivo seja alterado após uma destas referências. No diretório do sistema, no entanto, tal incremento sòmente será feito quando a referência fôr efetivamente iniciada por duas razões principais:

1. Se ocorre algum problema com o usuário e as referências não são executadas o diretório do sistema não precisa ser atualizado. (Havendo problema o diretório do usuário pode ser destruído sem causar prejuízo aos demais).

2. Os arquivos do sistema ficam com maior liberdade de atualização sendo feita a operação sempre com a última versão existente. (Se a operação tiver sido iniciada e o arquivo fôr atualizado, é claro que a operação prosseguirá com a versão já existente).

Operações tais como cópia e listagem (sòmente para arquivos do sistema) e descarregamentos e execuções (para todos os arquivos) devem ao terminar o processamento de cada arquivo decrementar o byte de referência. A procura no diretório, nêste caso será feita pelo "ponteiro do mapa do disco" que permanecerá imutável (note-se que o nome pode mudar).

REGISTRO DE RÓTULO:

O registro de rótulo é a imagem do rótulo de uma fita cassette. No próprio registro são mantidas indicações de quais os registros a serem lidos ou gravados.

O formato do registro de rótulo é o seguinte:

RESERVADO	IDENTIFICAÇÃO DO VOLUME	ARQUIVO 1	.....	ARQUIVO 19	ARQUIVO 20

Tamanho: 512 bytes

Campos:

2 B - Reservado: área reservada para futura extensão da identificação da fita (no momento, utilizada para identificação do registro de rótulo). Contém o número da fita em binário.

70 B - Identificação do volume: área mantida para identificação do volume.

440 B - Arquivos: área reservada para identificação dos arquivos mantidos neste volume. Capacidade: 20 entradas.

1) Identificação do Volume:

Formato:

NUMERO DE VOLUME	DATA CRIAÇÃO	DATA ÚLTIMA ATUALIZAÇÃO	RESERVADO	

LEITURA		GRAVAÇÃO		RESERVADO	Nº DE ARQ. (TOTAL)
Nº DE LEITURAS	Nº DE ERROS	Nº DE GRAVAÇÕES	Nº DE ERROS		

Tamanho: 70 bytes

Número de volume: número da fita (2B)

Data de criação: data em que foi inicializada esta fita (3B)

Data última atualização: data em que pela última vez o rótulo foi lido ou gravado (3B)

Reservado: 3 bytes reservados para implementação da data de liberação (3B)

Número de leituras: número de vezes que esta fita foi lida (2B)

Número de erros de leitura: número de vezes em que foi detetado erro de leitura (2B)

Número de gravações: nº de vezes que a fita foi gravada (2B)

Número de erros de gravação: nº de vezes em que foi detetado erro de - gravação (2B)

Reservado: 48 bytes reservados para futuras implementações, tais como:

- proteção de volume
- indicador de continuação de arquivo de outro volume
- formato de gravação
- tamanho do bloco ou registro
- palavra-chave de proteção, etc. (48 B)

Número de arquivos: número de arquivos já gravados neste volume (1B)

2) Identificação do Arquivo:

Formato:

NOME	FREQUENCIA DE UTILIZAÇÃO	POSIÇÃO	PROTEÇÃO E OPERAÇÃO	BLOCOS ALOCADOS	BLOCOS OCUPADOS	RESERVADO

Tamanho: 22 bytes

Nome: identificação do arquivo (4 bytes)

Frequencia de utilização: nº de vezes que o arquivo foi lido e/ou gravado até a próxima edição da fita. Serve para a otimização da edição - arquivos com maior frequencia de utilização serão colocados no início da fita (2 bytes).

Posição: posição do arquivo no volume (1 byte)

Proteção e Operação: (2 bytes)

00000000 - 1º byte: Proteção

1..... - arquivo protegido para gravação

.1..... - arquivo protegido para leitura (deve ser dada palavra-chave)

..1..... - arquivo concatenado

...1.... - arquivo não se encontra no disco

00000000 - 2º byte: Operação

1..... - arquivo sofrerá descarregamento

.1..... - arquivo sofrerá numeração

..1..... - arquivo será reservado

...1.... - arquivo será liberado na gravação

Blocos alocados: nº de blocos disponíveis para este arquivo (1 byte)

Blocos ocupados: nº de blocos efetivamente ocupados por este arquivo (1 byte)

Área reservada: 9 bytes

Esta área foi reservada para futura implementação de estatística sobre o arquivo, tal como número de linhas excluídas, incluídas, alteradas, etc.

Processamento de Registro de Rótulo

1) Tipos de rótulo

a) Normal: referência fita que foi lida por intermédio do comando CARREGUE.

b) Pseudo-rótulo (DUMMY-label) - rótulo que é criado no momento em que o usuário tecla o comando DESCARREGUE e a fita referenciada - não foi lida pelo sistema. O formato do P-R é idêntico ao do rótulo normal, apenas faltando os dados que não foram lidos do rótulo da fita. Os dados preenchidos pelo comando DESCARREGUE no P-R serão utilizados para a atualização do rótulo da fita no momento da gravação. O P-R, após a sua criação tem o tratamento - de um rótulo normal.

2) Leituras: São lidos os rótulos e arquivos especificados pelo comando CARREGUE, zerando-se os bits do campo operação, e marcando-se os arquivos que não estão no disco. Caso não haja espaço no disco para que seja descarregado algum arquivo da fita, deve ser emitida mensagem e o usuário - volta ao limbo de espera, não sendo carregado nenhum dos arquivos da fita.

3) Processamento do Usuário:

Os comandos descarregar e limpe são processados com o rótulo da fita, devendo ser emitida mensagem caso o arquivo não caiba ou não exista na fita. Neste caso, o arquivo será marcado como reservado. Serão emitidas mensagens de erro p/comando errado (arquivo não existente).

Caso seja necessária a edição nesta fita no momento da

gravação, esta informação deve ser guardada no diretório do usuário.

Os comandos execute e reserve são processados diretamente no diretório do usuário, desta forma economizando um acesso ao disco.

4) Gravação No momento da gravação é que serão verificados os erros existentes para comando descarregue que deu origem a um P-R. O procedimento adotado é o mesmo que para o rótulo normal, ou seja, caso não haja espaço e não seja possível a edição automática, o arquivo deve ser colocado como reservado (neste caso, a reserva é forçada).

Caso seja necessária a edição automática, deverão ser lidos da fita todos os arquivos que não se encontram no disco. Se isto não for possível, devido à falta de espaço em disco, o arquivo ou arquivos que provocaram a edição serão marcados como reserva forçada. Se for possível a edição, no novo rótulo os arquivos serão gravados em ordem crescente do campo frequencia de utilização, que será zerado.

OBSERVAÇÕES: Tanto na leitura quanto na gravação, deverão ser atualizados os campos correspondentes do rótulo, tais como:

número de leituras

número de erros

data última atualização

frequencia de utilização

estatísticas

etc.

ADMINISTRAÇÃO DA MEMÓRIA

Após o exame das diversas alternativas, decidiu-se, principalmente com o objetivo de economizar memória não armazenar texto dos usuários. Assim, cada usuário teria apenas um buffer de entrada/saída de 50 palavras capaz de acomodar comandos e linhas de texto. Cada linha, à medida que fosse referenciada seria trazida à memória e posta nêste buffer. Os cálculos a seguir avaliam a viabilidade desta alternativa.

Dados:

Velocidade do usuário:	1 caráter por segundo
Linha média:	20 caracteres
Velocidade do terminal:	10 caracteres por segundo

as requisições de disco irão variar segundo o serviço do usuário;

terminal em listagem:

pede um acesso (leitura) a cada 2 segundos

terminal em entrada:

pede um acêrto (ler, regravar) a cada 20 segundos

terminal em acêrto:

pede um acêrto (ler, regravar) a cada 20 segundos

Hipótese 1: todos os terminais listando

160 requisições em 10 segundos; carga do disco = 160 x 70 ms =

= 11200 ms/10000 ms = 112%

Tempo de Resposta (pior caso) = 32 x 70 = 2240 ms

Hipótese 2: todos em acêrto ou entrada:

16 requisições em 10 segundos; carga do disco =

= 16 x (70 + 40)ms = 1760/10 000 ms = 17,6%

Tempo resposta (pior caso) = 3520 ms

Hipótese 3: metade em cada modo:

88 requisições em 10 segundos;

carga do disco = 65%

Interferência dos cassettes:

Uma unidade de fita cassette em funcionamento gera, aproximadamente, 1 requisição por segundo assim sendo; 4 unidades em 10 segundos gerarão 40 requisições ocasionando, no disco, uma carga de 28%

Observação:

Uma vez que a maior carga é ocasionada pelos terminais em lista

gem, uma possibilidade de reduzí-la seria manter o buffer do usuário sempre cheio (de acôrdo com a linha-média poder-se-ia armazenar 4 linhas) e, antes de buscar uma linha (em listagem) verificássemos a presença da mesma. A carga do disco para esse caso poderia ser então reduzida para perto de 30%.

Análise do funcionamento do disco:

A seguir temos uma análise do funcionamento do disco (ainda sem computar os estouros).

admitamos metade dos terminais em listagem e metade em acêrto;

suponhamos ainda o funcionamento simultâneo dos 4 cassettes: sabemos que, nêste caso, podemos esperar em 10 segundos 128 requisições assim distribuídas:

80 de listagem

8 de acerto

40 de cassettes:

Tipo de req.	Nº/seg.	T.acesso	Latência	Rotação	Total Unitário	Total Geral
Listagem	8	50	20	-	70	560
Acêrto	1	50	20	40	110	110
Cassette	4	50	20	-	70	280
						960 ms

Carregando totalmente o buffer do usuário esta tabela seria da seguinte forma:

Tipo de req.	Nº/seg.	T.acesso	Latência	Rotação	Total Unitário	Total Geral
Listagem	2	50	20	-	70	140
Acêrto	1	50	20	40	110	110
Cassette	1	50	20	-	70	280
						530 ms

Interferência de "Estouros"

A cada ocorrência de estouro devemos somar o tempo previsto para o algorítmo escolhido. Parece razoável supor que (com boa margem de segurança) 20% dos acêrtos causarão estouro.

Dessa forma o tempo do atendimento de acêrto poderia variar entre 110 a 290 ms dentro de uma média de 170 ms. Ainda assim teríamos uma ocupação de 59% do disco.

Interferência de mensagens de êrro:

Frequentemente haverá requisições para localizar e imprimir mensagens de êrro. No entanto esta carga já está incluída nos acêrtos pois a

mensagem ocorre em resposta a um comando do usuário e é equivalente ao pedido de listar uma linha de texto.

Outras interferências:

Ainda devem ser consideradas as interferências causadas por operações de cópia, renumeração, apagar, transmissão para o computador etc. A frequência dessas operações já está incluída nos acertos; porém o tempo de execução de algumas destas operações difere do padrão.

V. MONITORAMENTO

INTRODUÇÃO

O software do sistema será dividido em duas partes: rotinas de atendimento a interrupções e rotinas de processamento, estas últimas também chamadas de PROGRAMÃO.

A idéia predominante do sistema é que todo o processamento seja feito pelo programão sem conhecimento de interrupções. As rotinas de interrupção são destinadas sòmente a tratar as interrupções e, eventualmente, atualizar ponteiros e outros parâmetros nas tabelas de informação de tal forma que o programão efetue o processamento apropriado quando estiver dando atenção a êste usuário.

Para melhor ilustrar o processamento, vamos usar a seguinte descrição aproximada:

Consideremos uma tabela de 32 terminais que, para simplicidade serão processados cíclicamente sem nenhuma diferença de prioridade.

Inicialmente, o programão começa a ser executado com todos os ponteiros referenciando o terminal nº 1. É feito todo o processamento possível para êste terminal até que o mesmo necessite aguardar um evento. (por exemplo ler uma linha de texto do terminal); nêste ponto, todos os ponteiros e parâmetros são mudados para fazer referência ao terminal nº 2. (O endereço do programão que deverá ser executado para o terminal nº 1 será armazenado na Tabela de Informação). O terminal nº 2 recebe processamento até esperar um evento quando o terminal nº 3 passará a ser atendido e assim sucessivamente.

Após atender ao usuário do terminal nº 32, novamente será atendido o usuário do terminal nº 1.

Note-se que não há quebras no processamento devido a interrupções. Se o terminal nº 23 estiver sendo processado e fôr completado um evento do terminal nº 11, a rotina de interrupção, após atender ao evento marcará que o terminal 11 não mais depende de evento e o processamento do terminal nº 23 continuará normalmente.

Findo o processamento do terminal nº 23 um nôvo terminal será escolhido para processamento (em nosso caso, o terminal nº 24) e o terminal nº 11 sòmente será atendido quando fôr sua vez (por exemplo, após atender ao terminal nº 10).

Na verdade o processamento não será cíclico pelas seguintes razões:

a. Terminais que aguardam eventos não recebem processamento.

b. A fila de usuários aguardando CPU será atendida na base "FIRST IN FIRST OUT".

No esquema proposto o programão será feito tendo em vista atender a um só terminal e, sendo um programa reentrante (podendo haver exceções em alguns trechos) poderá ser usado por todos os terminais.

As tabelas de informações (TIT,TIC) conterão os parâmetros necessários para se proceder ao processamento descrito acima. As seguintes informações contidas nestas tabelas são destinadas principalmente a êste - fim:

(8b) - ponteiro para o próximo (a ser processado) da lista

(16b) - endereço do programão

FUNCIONAMENTO:

Multi-Stack

Observação: esta alternativa é descrita ilustrativamente, uma vez que a mesma não será usada no sistema.

A multi-stack é uma área de memória destinada a armazenar endereços de retôrno para subrotinas do programão. A necessidade de armazenar os endereços fora do programa advém do fato que pelo menos as partes não contínuas em execução devem ser reentrantes.

Como solução alternativa poderíamos ter uma stack de 4 ou 5 palavras para cada usuário permitindo, portanto, até 5 níveis de subrotinas por usuário. A desvantagem nêste caso é que, possìvelmente, muita memória seria desperdiçada, uma vez que, estatisticamente, é de se esperar que enquanto alguns usuários possam estar no 3º nível de subrotinas, outros poderão estar no nível 0.

A multi stack se propõe a abrigar a stack de todos os usuários em uma área comum. Cada nó da multi stack contém 2 palavras: endereço de retôrno e apontador para o nó anterior. Os nós não usados são mantidos em uma lista de vazios.

Desvantagens no uso da MULTI-STACK:

São básicamente duas:

1. Uso do apontador (que não seria necessário se tivéssemos 1 stack por usuário).

2. Ligação de Subrotinas - que não poderá ser feita com as instruções comuns; para chamar uma subrotina tem-se que executar uma série de 5 instruções ao invés de sòmente uma instrução.

Supressão da MULTI-STACK

A supressão da multi stack só pode ser feita se pudermos garantir que a parada do programão sòmente se dê em nível 0 de subrotina. Certamente não há impossibilidade de se recorrer a esta alternativa. Porém deve-se pesquisar cuidadosamente a organização dos programas de modo a que não se faça uso ineficiente da memória para armazenar código.

Uma possível organização seria a seguinte:

Todas as rotinas capazes de gerar eventos teriam as seguintes restrições:

1. cada uma destas rotinas só pode gerar 1 evento efetivamente.
2. só podem retornar ao nível 0 de subrotinas.
3. Após execução de uma destas rotinas o contrôle não volta imediatamente ao chamados; o contrôle é entregue ao próximo usuário do ciclo.

Processamento:

Com a finalidade de manter o máximo de compatibilidade com os métodos usuais propõe-se o seguinte esquema de funcionamento:

- AO INICIAR O PROCESSAMENTO DE UM USUÁRIO

1. anotar posição atual da stack
2. desviar para programão

- APÓS PROCESSAR UMA ROTINA GERADORA DE EVENTOS

1. apagar a stack
2. executar endereço do programão

Pode-se ver que o esquema proposto funciona como se segue:

A ligação de subrotinas é feita normalmente usando a stack do sistema. A posição da mesma por ocasião do início do processamento é anotada. Esta posição, que pode ser a 1a. disponível da stack, armazenará o endereço de retôrno ao nível 0 de programa sempre que fôr efetuada uma chamada a subrotina. Após processar uma rotina geradora de evento, sabe-se,- pelas restrições impostas, que o retôrno não será feito à rotina chamadora, mas sim ao nível 0 (primeiro chamador). Isto envolve um retôrno para o endereço contido na 1a. posição da stack (cujo apontador foi anotado) e destruição da stack. É possível que, para indicar condições especiais, tais - como êrros etc. O retôrno ao programa chamador, (nível 0) seja feito para pontos diferentes.

Nêste caso o endereço de retôrno calculado pelas subrotinas - chamadas deverá substituir o endereço original contido na base da stack, o que poderá ser feito sem maiores problemas.

É interessante notar que o processamento de um usuário sempre termina em uma rotina geradora de evento.

Com o intuito de aumentar a eficiência do sistema (menor código, maior rapidez) optou-se por esta alternativa ao invés da MULTISTACK.

MUDANÇA DE CONTRÔLE

O contrôle do programão, como já foi visto será dado alternativamente a cada um dos usuários do sistema. A seguir é descrito o procedimento proposto para mudança de contrôle; isto é, transferir o contrôle do programão de um usuário para outro.

A escolha do próximo usuário e o mecanismo necessário ao processamento também são descritos.

Listas

Dentro da Tabela de Informações serão mantidas diversas listas usando o campo "apontador para o próximo".

Essas listas serão denominadas:

1. aptos a processar - lista dos usuários que não aguardam nenhum evento.
2. aguardam liberação de disco - lista dos usuários que dependem de - inicializar uma Entrada/Saída para o disco.
3. aguardam resposta do usuário - lista dos usuários que dependem de ações do próprio usuário para processar.
4. aguardam outros eventos

Denominaremos "usuário em processamento" (U.P.) ao usuário que detém a CPU e "usuário do disco (U.D.) ao usuário que está sendo atendido pela rotina de disco.

Premissas

Supondo que o disco é um recurso crítico no sistema, foi observada a necessidade de mantê-lo constantemente ocupado. Assim sendo, uma vez completa uma operação de disco, outra operação deverá ser inicializada o mais breve possível

Operações Físicas e Lógicas (DISCO)

As operações em disco têm dois tipos de término:

Término físico quando uma operação (comando de E/S) do disco (ex. ler um setor, posicionar braço, etc.)

Término lógico quando a operação solicitada pelo usuário está completa (regravação de um setor, acesso a uma linha etc.)

vê-se que para uma operação lógica podem ser necessárias uma ou mais operações físicas.

Atuação das Rotinas de Interrupção

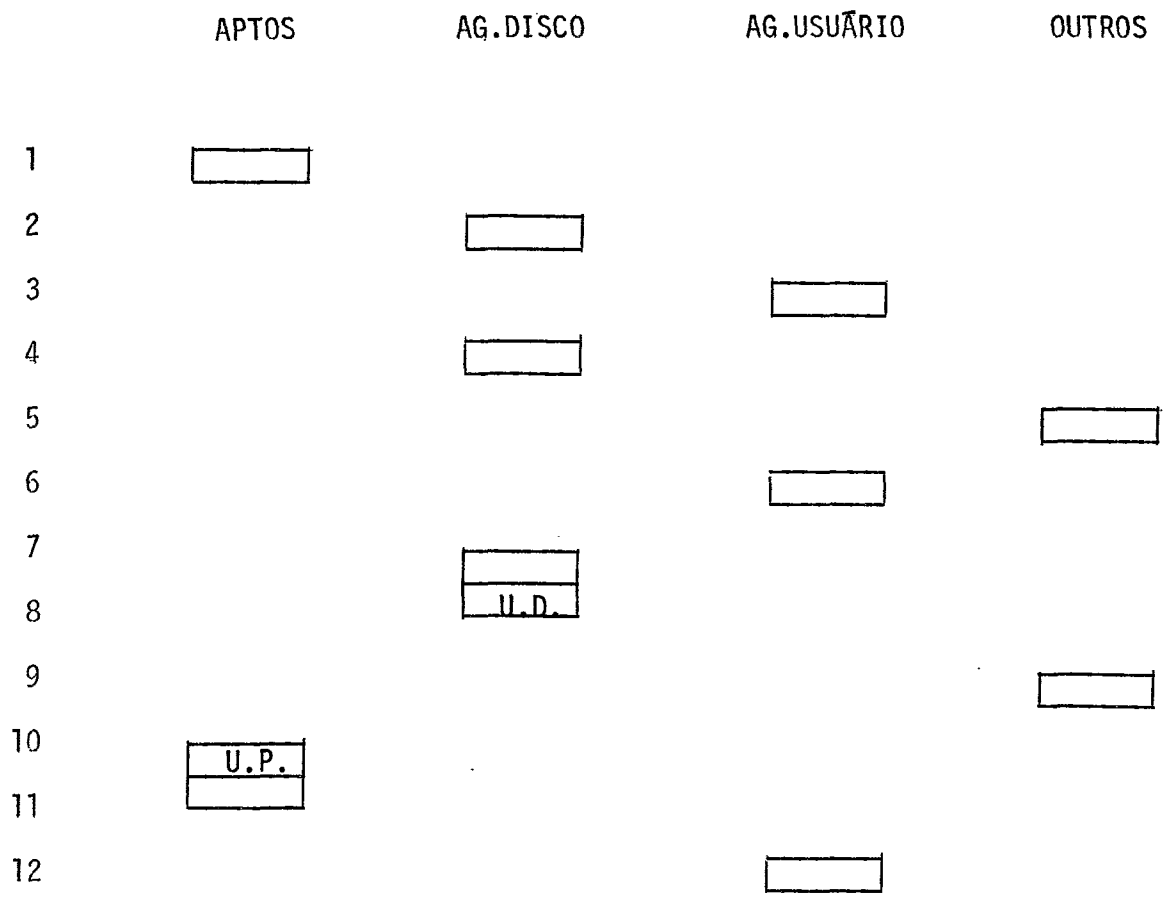
As rotinas de interrução ao detetarem o fim (lógico) de uma operação devem providenciar para que o usuário seja deslocado para a lista de "aptos a processar".

A rotina de disco, além de providenciar êste deslocamento, examinará a lista dos que "aguardam disco" e, de acôrdo com o critério exposto adiante (Prioridade de operações em disco) escolherá o nôvo "usuário do disco";

Como ilustração podemos ver as figuras 1, 2 e 3

Na figura 1 (sistema com 12 usuários) U.D. = 8 e U.P. = 10 isto é, está sendo processado o usuário nº 10 e o disco está servindo ao usuário nº 8.

F.1



U.D. = 8

U.P- = 10

- Quando a operação (lógica) do usuário 8 termina, a rotina de interrupção desloca êste usuário para a lista de "aptos" e percorre a lista de"aguardam disco" para escolher o próximo U.D. Suponhamos que seja o usuário nº 4.

F.2

	APTOS	AG.DISCO	AG.USUÁRIO	OUTROS
1				
2				
3				
4		U.D.		
5				
6				
7				
8				
9				
10	U.P.			
11				
12				

U.D. = 4

U.P. = 10

A figura 3 mostra a mudança de contrôle (feita sempre fora de interrupção). O usuário 10 fica bloqueado por um evento (por exemplo, disco) e é pôsto na lista apropriada. O próximo usuário da lista de aptos (no caso o nº 11) recebe o contrôle do programão.

	APTOS	AG.DISCO	AG.USUÁRIO	OUTROS
1				
2				
3				
4		U.D.		
5				
6				
7				
8				
9				
10				
11	U.P.			
12				

U.D. = 4

U.P. = 11

Observe-se que a lista de aptos não necessàriamente obedece a ordem sequencial das tabelas de informações.

Prioridade de Operações em Disco

Uma vez finda uma operação lógica do disco a lista de "aguardam disco" será varrida e será escolhido um nôvo "usuário do disco".

Uma solução seria pegar o primeiro da fila (o que aguarda há mais tempo) sem se preocupar com nenhuma otimização. Outra solução interessante seria atender ao pedido mais próximo da posição do braço. Êste critério, que tem sido usado com bastante sucesso em disco de cabeça fixa não parece adequado para discos de cabeça móvel porque um pedido de acesso a uma área pouco usada do disco sòmente seria atendido quando não mais houvesse pedidos pendentes ou seja duraria um tempo indeterminado. (No disco de cabeça fixa a proximidade é devida à rotação e, portanto varia em função do tempo).

Uma adaptação que nos parece válida é atender ao pedido mais próximo, observando-se o sentido de movimentação do braço; ou seja, o sentido de movimentação do braço de leitura pode ser para o início ou para o fim do disco. O próximo pedido a ser atendido será aquele que esteja mais próximo da posição atual e o deslocamento do braço seja feito no mesmo sentido (como se fôsse um elevador). Caso não exista pedidos no mesmo sentido, êste será invertido.

Êste algorítmo se propõe a uma otimização no acesso garantido, no entanto, um limite no tempo de atendimento, sem ser demasiadamente complexo.

ARRUMAÇÃO DAS LISTAS

Cada uma das listas possui uma ordem que pode ser mais conveniente para seu processamento. A seguir analisamos estas ordens e a possibilidade de mantê-las.

"aptos a processar" - não há ordem primordial para esta lista
- os usuários seriam postos na lista sempre pelo fim - da mesma. Assim seria processado em 1º lugar o usuário que estivesse há mais tempo aguardando processamento. Esta ordem é a mais conveniente para colocar os usuários na lista.

"aguardando disco" - a melhor ordem seria dada pelos endereços de disco - crescentes ou decrescentes) pois teria a grande vantagem de facilitar a procura do próximo "usuário do disco". Além do mais, se mantivermos o U.D. nesta lista, o próximo U.D. será encontrado imediatamente. A colocação na lista pode ser mais demorada por exigir uma varrida parcial na mesma porém a determinação do próximo U.D. (que será feita durante uma interrupção) será muito mais rápida pois em outro esquema seria necessária uma varrida total da lista.

"aguardando usuário" - a melhor ordem para esta lista seria dada pela "hora da última interrupção" uma vez que esta lista serve, primordialmente, para forçar o TCHAU. Se considerarmos o tempo de tolerância como intervalo entre a entrada de dois caracteres esta implementação exigirá muito overhead. Se, no entanto o tempo de tolerância for considerado o intervalo entre o sistema aguardar o usuário e êste pressionar o retôrno co carro, então tal manutenção será extremamente simples, bastando colocar cada usuário no fim da fila.

"outros eventos" - não há ordem primordial uma vez que nenhuma procura está prevista nesta lista. A única aplicação possível seria detetar eventos com tempo de resposta anormal e, consequentemente, panes de equipamento ou do sistema. Nêste caso seria conveniente a ordem natural da geração dos eventos cuja colocação (sempre no fim da lista) seria bastante simples. A retirada (como é em ordem aleatória) seria feita mais rápidamente usando um ponteiro inverso podendo, também ser dispensado em troca de se varrer a lista.

Se não fôr implementado êste contrôle de eventos uma solução seria deixar êstes usuários numa lista "desconexa" (sem apontadores) uma vez que nenhuma procura seria feita.

BIBLIOGRAFIA

1. ON LINE TEXT EDITING, A SURVEY - Andries Van Dam e David E. Rice - ACM Computing Surveys Vol 3 Nº 3 Setembro de 1971 - descreve comparativamente vários sistemas editores de texto.

2. WYLBUR: An Interactive Text Editing and Remote Job Entry System Roger Fayman e John Borgelt Communications of the ACM Vol 16 Nº 5 - Maio de 1973 - Descreve resumidamente o projeto do Editor Wylbur da Universidade de Stanford.

3. IBM System /360 OS Conversational Remote Job Entry. Terminal User's guide. GC-30-2014. Êste manual descreve a utilização dêste editor da IBM.

4. WYLBUR reference manual - Computer Facility Stanford University Computing Center. Descreve o editor WYLBUR para os usuários.

5. PDP-11 - Processor Handbook - Digital Equipment Corporation, Maynard, Massachussetts. Descreve o computador PDP-11 utilizado para implementar o projeto.

6. Pretexto, Manual de Desenvolvimento - Documentação Interna - NCE-UFRJ
   Descreve o sistema proposto nêste mesmo nível de detalhe sendo porém uma documentação atualizável.

7. Pretexto, Manual de Implementação
   NCE-UFRJ - Documentação Interna (atualizável)
   Descreve os procedimentos de implementação do sistema

8. Pretexto - Manual de Especificações
   NCE-UFRJ - Documentação Interna (atualizável)
   Descreve todos os algorítmos de programas do sistema