PCS 3446 – Sistemas Operacionais

Aula 26 – Estudo de Caso – Sistema P

Fonte:

L. S. Keller, *Operating Systems* – cap.9

1. Sistema Operacional Universal?

- A tarefa de escolher o sistema operacional é não é trivial
- Será que essa seleção poderia ficar mais simples através da adoção de um único sistema operacional padrão, deixando apenas a arquitetura do computador para ser avaliada e selecionada?

- Problemas com essa meta utópica
  - Como poderiam os implementadores de um sistema operacional universal assegurar que todas as instâncias sejam funcionalmente idênticas , não só compatíveis?
  - Como poderia um sistema universal ser instalado em todas as máquinas com diferentes tamanhos de palavra?
  - Estabelecidos os requisitos, como seria tal sistema operacional capaz de atender a todos eles?
  - Com o tempo, as exigências mudam. Que fazer quando o sistema único deixar de ser universal?

- O sistema P da UCSD aqui discutido tentou ser uma espécie de sistema operacional universal
- Ele conseguiu solucionar os problemas de identidade funcional e da diferença de tamanho de palavra escondendo a sua causa, ou seja, as arquiteturas dos computadores reais
- O sistema operacional P foi projetado e construído para uma pseudo máquina ideal, com pseudo arquitetura ideal
- Essa inovação tem hoje o nome de máquina virtual
- A letra P em “sistema P” abrevia o termo “pseudo”

- A pseudo-máquina possui sua própria linguagem de máquina p-code.
- O sistema P é um programa em P-code que pode ser executado em qualquer p-machine
- Essa técnica não apenas facilita a obtenção da portabilidade mas é produtiva, já que propicia implementar o sistema P uma única vez
- Os reais fatores que na prática irão distinguir entre si as máquinas que executam o sistema P serão o desempenho, o custo e a confiabilidade

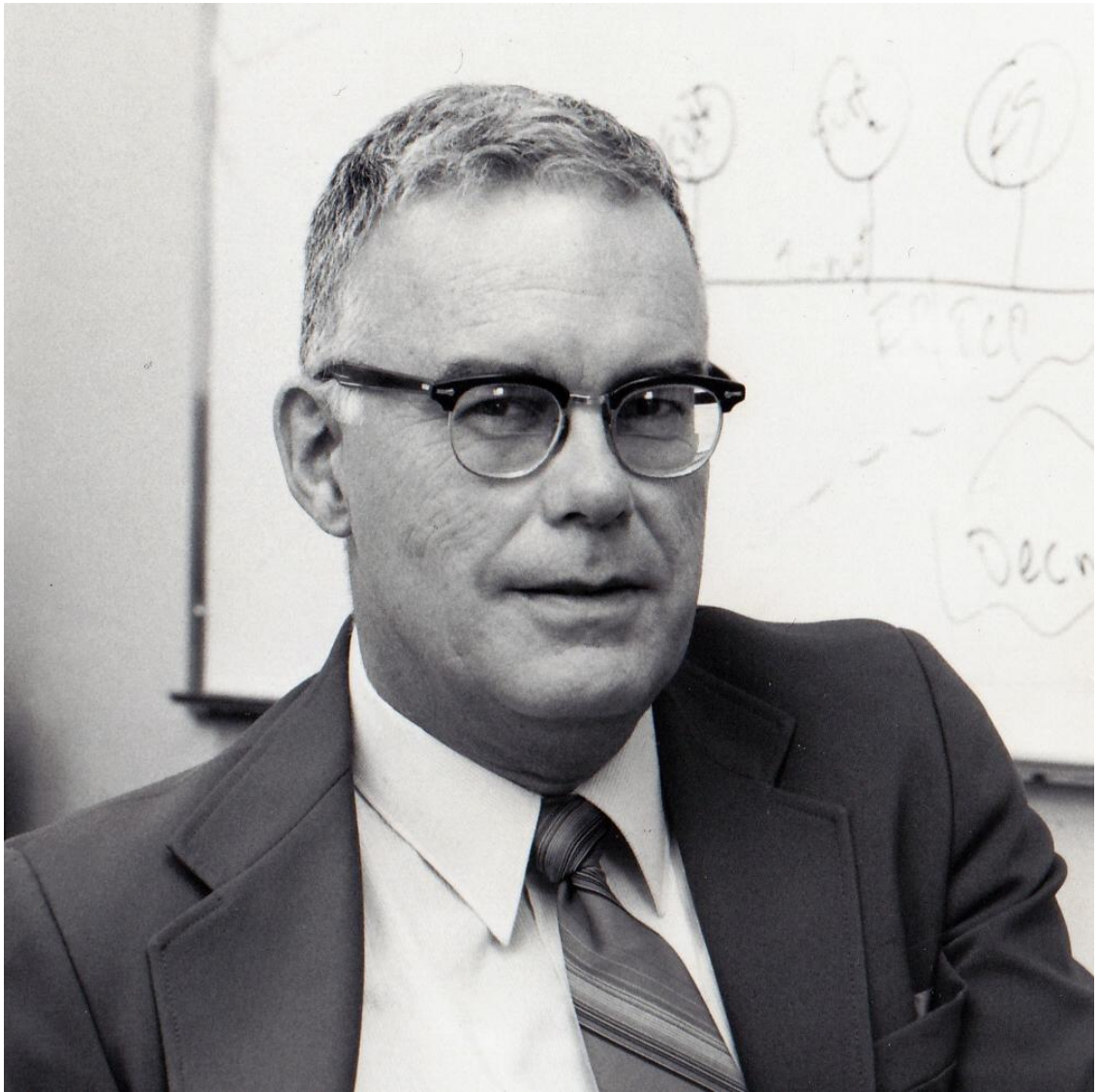
- Neste estudo de caso vamos examinar a arquitetura da p-machine, os recursos fornecidos pelo sistema P e a forma como esses recursos chegam ao usuário
- Será mostrado também como a p-machine é implementada em máquinas reais, em particular algumas dificuldades encontradas e a forma como foram superadas
- O sistema operacional e a sua relação com o simulador da p-machine através da linguagem de programação UCSD Pascal também serão discutidos

2. Histórico

- A história do sistema P está intimamente relacionada com o desenvolvimento da linguagem de programação Pascal e com o advento dos microcomputadores
- O desenvolvimento da linguagem Pascal nos primeiros anos da década de 70 foi um importante marco na computação
- A linguagem fornecia recursos de apoio à técnica então emergente da programação estruturada

- Pascal foi a linguagem que em primeiro lugar angariou adeptos por causa da sua aderência a essa técnica, facilitando ensino da linguagem
- Na universidade da Califórnia em San Diego (UCSD), Pascal estava sendo usada em um sobrecarregado sistema de time-sharing, quando se tomou a decisão de migrar para os microcomputadores, então recentemente disponibilizados,.
- Em 1974 um grupo de estudantes, trabalhando sob a orientação de Ken Bowles começou o projeto do qual resultou o sistema P.

Kenneth Bowles, UCSD, 1974, criador do UCSD Pascal



- Certas decisões técnicas fundamentais foram tomadas logo cedo durante este projeto
  - O grupo decidiu que seria necessário desenvolver todo um ambiente de desenvolvimento para o Pascal. Iisso envolvia o sistema operacional, editor, compilador e administrador de arquivos
  - Decidiram que este software deveria ser fortemente integrado, estabelecendo harmonia entre partes no momento não disponíveis , dado que os sistemas eram desenvolvidos de uma forma fragmentada
  - Decidiram também que o sistema deveria ser atraente tanto para o estudante novato como para o programador mais experiente
  - Diferente de muitos sistemas existentes àquela época, escolheram para o terminal do usuário uma tela de vídeo e não uma máquina de escrever
  - Como um padrão para o Pascal ainda não havia surgido, o grupo julgou aceitável estender a linguagem para que pudesse atender requisitos para aplicação em ensino e na programação de sistemas
  - Devido a essa relação muito próxima entre o sistema p e o Pascal torna-se conveniente ter uma ideia ao menos superficial do aspecto dos programas Pascal UCSD

- De forma crucial uma vez que o grupo uma explosão no mercado de microcomputadores de baixo custo eles não quiseram fazer nenhuma decisão prematura acerca do hardware
- decidiram projetaram em torno de uma máquina virtual uma pseudo máquina
- essa decisão se provou boa sucesso que teve em facilitar a produção de compiladores Pascal um único compilador Pascal fosse criado, com a ajuda do qual fosse produzido p-code para uma máquina virtual.
- Isso exigia apenas a produção de geradores de código que fossem capazes de traduzir p-code para código de máquina de um particular hardware real
- As primeiras três decisões resultariam em um sistema de software relativamente grande comparados ao tamanho usual de memória de um microcomputador típico daquela época (48 a 64 kilobytes)
- Os compiladores para linguagem Pascal deveriam produzir programas objeto grandes por causa do baixo nível do conjunto de instruções presentes nos microprocessadores disponíveis

- Assim, a técnica da interpretação foi preferida: em vez de traduzir p-code para código de máquina, como fazem os compiladores, o interpretador executa o programa à medida que o traduz.
- O código de máquina que executa isso é, de fato, parte do código do interpretador eu código particular executado depende do software do interpretador.
- O particular código processado é determinado pelas instruções p-code que vão sendo encontradas nessa operação.
- Traduzindo programas Pascal para um p-code compacto, seguido da interpretação esse código, e não de sua tradução, a implementação pode adquirir considerável ganho em eficiência.
- Usando essa técnica, a equipe do projeto rapidamente produziu adaptações bem-sucedidas do sistema Pascal para vários processadores: pdp-11, LSI-11, Z80 e 8080.

- O sistema P em sua versão 3 da Western Digital no Pascal Microengine interessante por quê essa última se transformou em uma máquina real que executa p-code como sua linguagem de máquina
- Adicionalmente a Western Digital versão 3 também incorporou recursos de concorrência em sua versão do UCSD Pascal, com forte impacto no desenvolvimento da versão 4 do sistema
- Em 1985 as versões 4.2 e 4.21 foram liberados com grandes melhorias: no desempenho, nos recursos da linguagem de programação, e o que é mais importante, em sua capacidade de operação em rede

3. Uso de comandos de menu

- Tipicamente, o sistema p é executado em sistemas de microcomputador cuja configuração contendo:
  - Memória primária, com pelo menos 64 kilobytes
  - Armazenamento secundário de acesso direto, consistindo tipicamente de ao menos 2 discos flexíveis, com capacidade de 320 kbytes cada
  - Monitor de vídeo capaz de efetuar scroll e movimentos do cursor para qualquer ponto da tela
  - Uma impressora
  - É possível alterar essa configuração, usando discos de menor capacidade ou então uma única unidade de disco

- Ao ligar o sistema, o usuário recebe um prompt para que efetue a entrada de um comando
- Esse prompt aparece no topo de uma tela e apresenta alguns dos comandos disponíveis nesse nível mais alto do comando
- Por exemplo, para entrar no gerenciador de arquivos, o usuário deve teclar f
- Se um comando exige parâmetros, o sistema os solicita através de outro prompt
- Isso evita ao usuário aprender a sintaxe de comandos complexos

- Assim para compilar um programa usuário deve primeiro teclar C
- O sistema P apresenta então um prompt solicitando o nome do arquivo que contém o programa fonte
- a vantagem desse tipo de comando é que favorece ao novato visualizar todas as possibilidades simplesmente olhando para os nomes dos comandos apresentados na tela sem impedir que um programador experiente pode digitar o comando completo através da inserção de uma série de letras isoladas, em rápida sucessão

4. O sistema de arquivos

- As restrições das arquiteturas dos microcomputadores para os quais as primeiras adaptações foram feitas resultaram em um sistema de arquivos simples mas eficiente para o sistema P
- Conceitualmente, arquivos são armazenados em volumes de dados
- Esses volumes são associados a vários dispositivos que podem ser classificados como: estruturados em blocos, discos flexíveis e discos rígidos; interativos, ou seja, terminais; ou um dispositivo apenas de saída, como é o caso de uma impressora
- Cada dispositivo é numerado, portanto pode ser referenciado por seu número.
- Mais importante: cada volume tem um nome de 7 caracteres
- Volumes de discos são dispositivos estruturados em bloco.
- Eles se compõem de uma matriz de blocos de 512 bytes de 8 bits
- Esses blocos correspondem usualmente a um setor do disco físico

- Os diretórios do sistema de arquivos são aderentes a essa estrutura lógica matricial
- A **estrutura lógica de um disco** do *p-system* inclui:
  - Área 1 (toda a trilha 0, opcional);
  - Área 2
    - *Bootstrap* ou teste (1K bytes);
    - Diretório (2K bytes);
    - Restante do disco lógico (arquivos do usuário);
  - Área 3 (parte do disco não usado, opcional)
- Os arquivos são armazenados em blocos contíguos. Isso propicia a ocorrência do fenômeno de fragmentação no espaço em disco
- O comando CRUNCH aplica garbage collection a um disco fragmentado
- Apesar da fragmentação, esse método de armazenamento fornece um acesso muito eficiente aos arquivos
- Abrindo-se um arquivo A, a ser lido sequencialmente, por exemplo, o sistema de arquivo necessita apenas verificar no diretório qual é a posição inicial e o comprimento do arquivo; o arquivo pode então ser acessado bloco a bloco; o sistema aloca o maior espaço possível (critério worst-fit) por default, mas é possível alocar metade do maior espaço disponível.

4. O sistema de arquivos

- Para completar a especificação de um disco é necessário fornecer a identificação do volume de mídia no qual ele está armazenado
- O nome do volume pode ter até 7 caracteres de comprimento e o nome do arquivo, até 15 caracteres
- Uma eventual sequência de caracteres após o ponto mais à direita simboliza o tipo de arquivo
- O sistema reconhece um certo número de tipos de arquivo, com frequência, arquivos de texto e arquivos de código
- Quando o sistema P é ligado, o disco que foi usado para dar boot no sistema, e que contém o interpretador e o sistema operacional é designado como sendo o volume raiz, podendo ser abreviado como *.

5. Arquitetura da Máquina P

- Até esse ponto, consideramos apenas a forma como o sistema P se apresenta para o usuário, os recursos que ele oferece e a interface com o usuário de que se utiliza
- Antes de mostrar o que está por baixo de tudo isso, é útil comentar da estrutura de um programa em UCSD Pascal
- A linguagem Pascal está tão fortemente acoplada ao sistema P que se torna difícil discutir um dos dois sem considerar também o outro.
- Os comandos de um programa Pascal correspondem aos encontrados na maioria das outras linguagens de programação: atribuições, condições, repetições, sequências, chamadas de procedimento
- O mais relevante para o nosso estudo nesta disciplina é o tipo de código que o compilador gera a partir dos comandos desse programa

- Um compilador Pascal escrito para uso em uma máquina real em geral produz código para ser executado por esse processador
- Por exemplo, um comando de atribuição, tal como i:=i+j, é traduzido para uma sequência de várias instruções dessa máquina
- Tais instruções tipicamente transferem valores da memória principal para registradores, onde operações de adição são executadas, e de onde o resultado obtido é transferido de volta para a memória principal
- A máquina P não utiliza registradores dessa maneira, mas em lugar disso emprega uma pilha. Pilha é uma estrutura de dados que implementa uma fila last in, first out.
- Assim, o compilador Pascal UCSD gera, para esse comando: p-codes para colocar os valores de i e de j no topo da pilha; um p-code de adição para executar a soma, eliminando as parcelas e preservando o resultado no topo da pilha; finalmente, o resultado seria removido do topo da pilha para ser depositado em i.

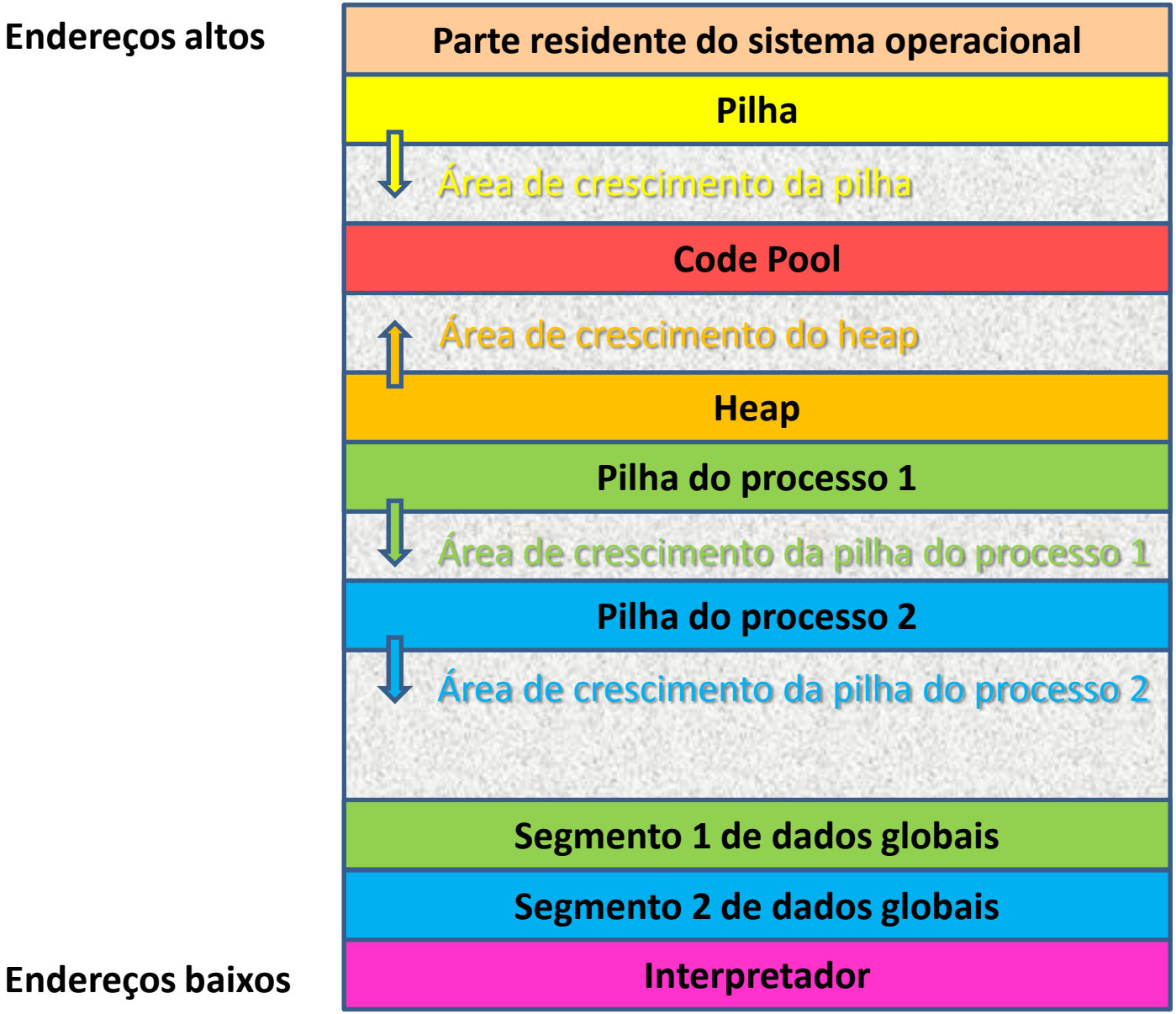
- Executar p-code em uma máquina real requer um emulador da máquina P que implemente uma pilha e o conjunto de operações aritméticas sobre essa pilha
- Um p-code que adicione inteiros efetua a operação de adição usando os valores contidos nas duas últimas posições do topo da pilha, e substituindo-os pelo resultado da operação, enquanto for possível resgatar valores previamente aí depositados, e guardando o resultado final em i.
- O p-code do programa do usuário (ou parte dele) deve estar presente na memória principal para que o emulador da máquina P consiga descobrir a próxima instrução da sequência do p-code a ser executada para que a ação apropriada seja efetuada.

- É importante notar que, a exemplo da linguagem Algol, da qual descende, Pascal é uma linguagem orientada à pilha.
- Todas as estruturas de dados estáticas de um programa (ou seja, aquelas cujos comprimentos são conhecidos na época da compilação do programa) são mantidas em uma pilha
- À medida que os procedimentos são chamados, seus dados locais vão sendo colocados na pilha, para daí serem removidos quando do retorno dos procedimentos ao ambiente que os ativou
- Por outro lado, o heap é mais apropriado para a construção das estruturas dinâmicas, cuja variação imprevisível de requisitos de espaço dificilmente pode ser antecipada

Segmentos

- Outro importante conceito incorporado ao Pascal é relevante para nossa discussão: os segmentos.
- Procedimentos declarados como segmentos não precisam estar obrigatoriamente presentes na memória principal para poderem ser executados (uma grande novidade na época)
- Caso não estejam, serão trazidos do disco e se na ocasião a memória principal estiver lotada, o espaço ocupado por um procedimento que não estiver sendo executado poderá ser resgatado como espaço vazio
- De fato, um programa em p-code é ele próprio um segmento, logo, arquivos de código se organizam em coleções de segmentos
- Notar a semelhança conceitual entre os segmentos e as bibliotecas de ligação dinâmica dos sistemas modernos (DLL – dynamic link library), as quais são incorporadas ao programa que as referencia apenas na ocasião em que sua execução for de fato requisitada.

Organização da memória da UCSD p-machine



- A pilha pode se expandir à medida que os procedimentos vão sendo chamados
- O heap pode crescer em direção à pilha enquanto estruturas dinâmicas vão sendo construídas no espaço vazio disponível
- Entre as áreas de pilha e de heap fica a chamada “code pool”, que é a área destinada ao crescimento de código no qual residem segmentos ou unidades de p-code
- As pilhas de processos mostradas como partes do heap são pilhas extra para áreas de dados de procedimentos especiais chamados processos, que podem ser executados concorrentemente

- A concorrência no sistema P é obtida usando o modelo de semáforo de Dijkstra, empregado na implementação do sistema operacional
- Segmentos globais de dados armazenam estruturas de dados que são globais aos programas
- Em outras palavras, eles ficam disponíveis para todos os procedimentos do programa
- Os segmentos de dados globais incorporam as estruturas de dados globais do sistema operacional

- O uso do Code Pool provê um sistema de gerenciamento de memória primária poderoso, embora ocasionalmente lento
- O Code Pool é organizado como uma área contígua da memória principal, no qual são alocados os segmentos do sistema operacional e dos programas do usuário
- Nessa área, os segmentos são conectados entre si na forma de uma estrutura de lista ligada

- Seu conteúdo pode ser alterado das seguintes formas:
  - Uma tentativa de expansão da pilha não é bem sucedida: isso é um Erro de Pilha, e a área é movida para baixo em direção ao heap, a fim de liberar espaço
  - Uma tentativa de expandir o heap fracassa: é o chamado Erro de Heap, e a área é movida para cima em direção à pilha
  - Uma tentativa de abrir espaço após um Erro de Pilha ou de Heap não tem êxito, e um ou mais segmentos são descartados sendo os segmentos restantes agrupados, para então tentar-se novo posicionamento
  - Não se consegue abrir espaço descartando segmentos e reposicionando: um erro fatal ocorreu no sistema (stack overflow) e o sistema precisa ser reiniciado

- Esses erros ou falhas do gerenciamento de memória primária são tratados por um processo subsidiário ao sistema P
- Esse processo tratador de erro não pode ele mesmo originar novos erros, porque seu espaço de pilha é alocado permanentemente no heap, sendo responsável apenas pela ativação de partes do sistema operacional permanentemente residentes na memória

- O tratamento de erros funciona da seguinte forma:
  - Caso o emulador da máquina P detecte um Erro de Pilha ou um Erro de Segmento (uma chamada feita a um segmento que não esteja na memória principal) o processo de tratamento de erro é ativado.
  - Ele processa o erro detectado manipulando o Code Pool ou, no caso de Erro de Segmento, trazendo para a memória o segmento referenciado.
  - Devolve então o controle ao interpretador de p-code, e reexecuta a instrução que havia originado o erro
  - Erros de Heap são detectados pelo sistema operacional, e este ativa a rotina associada de tratamento de erros

- Analogamente ao que ocorre com outros processadores, a p-machine possui registradores que são uma parte fundamental de sua arquitetura
- Não são registradores de propósito geral, mas são empregados para controlar a eficiência da p-machine
- São usados para executar funções tais como apontar o próximo p-code a ser executado, apontar o procedimento corrente, apontar o topo da pilha etc.
- A pilha é utilizada para computação em geral

6. Execução de programas em p-code

- Para executar programas em p-code nessa arquitetura com pilha, arquivos de código devem conter informação suficiente para que o emulador da máquina virtual P seja capaz de colocar seus dados locais na pilha, acessá-los para efetuar operações, e remover esses dados da pilha ao sair do procedimento
- O sistema precisa também estar apto a coletar todas as informações de que precisa cerca dos segmentos que compõem o programa
- Assim, um arquivo de código deve conter segmentos.
- De fato, inclusive um dicionário de segmentos é mantido aí, conforme mostra a figura 9 12

- Segmentos contém o nome de cada segmento em um programa e Indica onde o código deve ser armazenado
- Segmentos alinhados nas limites do bloco Apesar de que isso aumenta o tamanho dos arquivos de código primeiramente meios de acessar rapidamente o dicionário de segmentos Rios sedimentos eles próprios
- Cada segmento contém informação suficiente para o manuseio da pilha o formato de um segmento é mostrado na figura 9.13

- O ponteiro para dicionário de procedimentos na parte inferior da figura aponta para uma matriz de ponteiros (o dicionário de procedimentos) para o código associado a cada procedimento do segmento
- O código do procedimento contém p-codes implementam qualquer tipo de computação a ser executado naquele procedimento o espaço exigido para os dados locais é um ponteiro EXITIC para o código que deve ser executado assim que o procedimento terminar
- Quando um procedimento é chamado um registro de ativação é gerado e colocado no topo da pilha
- O registro de ativação contém para os parâmetros dados locais e informações de manutenção para a transferência de controle para um procedimento e retorno desse procedimento

7. O emulador da máquina P

- O emulador da máquina P pode ser considerada como sendo formado logicamente de duas partes :
- O interpretador, que faz a busca de cada p-code e seus parâmetros e executa a ação correspondente na pilha
- O pacote de suporte ao ambiente de execução time support Package , o qual executa operações em nível de máquina tais como mover ou discar sessões de área de memória movendo segmentos no Code Pool executando entrada e saída de baixo nível
- Uma parte do Run time support package, em particular é bem organizado para facilitar a portabilidade - a parte responsável pela entrada e saída, RSP/IO
- Esse é um componente praticamente Independente de máquina a menos de uma parte: BIOS – Basic input/output subsystem.
- O Bios varia dependendo dos dispositivos periféricos que estejam sendo usados mas a interface entre RSP/IO e BIOS é padronizado. Logo, existe uma hierarquia nesse processo de entrada/saída

- Um comando de entrada e saída do programa é traduzido para uma chamada de uma rotina de biblioteca, RSP/IO, (runtime support package I/O) a qual converte os parâmetros para montar a chamada apropriada de rotina BIOS, com os devidos parâmetros
- O uso de uma BIOS para implementar entrada/saída de baixo nível foi decidido uma fase bastante inicial do desenvolvimento do sistema P
- O primeiro pdp-11 não precisou dessa estratégia de portabilidade, mas a adaptação para Z 8080 usaram CP/M BIOS na execução de entrada/saída de baixo nível

- Quando se adaptou o sistema para outros processadores, ficou claro que seria necessário construir para eles algo similar ao BIOS do CP/M
- Outro importante auxílio à portabilidade foi o desenvolvimento de um SBIOS (simplified BIOS)
- Implementar uma BIOS não é uma tarefa simples exige que o sistema operacional já esteja implantado no novo hardware
- Assim, o chamado sistema adaptável pode ser usado para produzir rapidamente uma adaptação preliminar do sistema

- Isso se pode obter usando uma BIOS que possa fazer chamadas do SBIOS, de fato portanto acrescentando uma nova camada à hierarquia antes mencionada.
- Portanto, o implementador necessita apenas escrever uma SBIOS
- Uma vez que o sistema já esteja provisoriamente operando sobre essa SBIOS, ele pode ser usado para desenvolver, agora para a BIOS, outra adaptação mais eficiente.

8. Superando fraquezas e restrições

- Uma das objeções mais comuns ao sistema P é a de que, por ser o código interpretado, ele se torna lento
- Isso é verdade, entretanto o desempenho de um sistema de computação é medido não apenas pela sua potência computacional.
- É crucial para isso também seu throughput de dados
- Enquanto máquinas com código nativo muito eficiente podem superar facilmente os sistemas P interpretados, quando ambos são cpu-bound (processamento intenso) não é em geral tão grande a diferença de desempenho para códigos relativamente I/O bound

- Adicionalmente, programas em código nativo são necessariamente maiores do que aqueles escritos em p-code já que existem interpretadores que implementam no p-code recursos de alto nível do Pascal, quanto os códigos de compiladores nativos precisam produzir grandes quantidades de código de máquina para sintetizar os recursos desejados
- O tamanho grande do código nativo relativo ao p-code pode significar que o overhead da transferência de código nativo entre a memória secundária e primária pode degradar o desempenho global
- Isso é um fator importante no caso do gerenciamento da memória primária usado no controle de overlays.
- Isso ocorre na implementação de procedimentos do tipo segmento, no Pascal UCSD

- Geradores de código nativo convertem p-code para código de máquina nativo, e foram construídos para muitos processadores mas o seu uso impõe algumas penalidades.
  - O código Nativo de um programa costuma ser maior do que uma versão em P-code
  - Quando o interpretador transferir o controle ao código nativo ele deve salvar os seus dados trabalho, que vão ser restaurados quando o controle retornar a ele.
  - Esse chaveamento de contexto é ele próprio causador de overhead, no entanto novos geradores de códigos mais eficientes estão aparecendo
  - A única fraqueza mais séria do sistema é que o seu código é endereçado por bytes quando o tamanho da palavra da 16 bits permite endereços de até 64 kilobytes
  - Embora aplicações grandes possam ser acomodadas, um número excessivo de manipulações do code pool pode ser necessário.
  - Além disso, o desenvolvedor não pode tirar vantagem direta da quantidade maior de memória primária disponível em 16 e 32 bits.
  - Uma solução parcial mas eficaz foi a de separar áreas de dados (pilha e heap) em regiões fisicamente separadas da área de código.

- A configuração da figura 9.15 é chamada code pool externo diferente dos arranjos internos mostrados na figura 9.11

- Como os Erros de Pilha e de Heap no arranjo externo causam ainda mais Swapping o desempenho pode ser melhorado
- Code pools Multiplos também são possíveis mas o problema real é o acesso ao armazenamento de dados
- Um bom uso pode ser feito de uma área de memória primária grande: um volume de disco pode ser simulado na memória primária, e isso é chamado RAM-disk
- No momento do boot inicial todos os usuários do sistema são transferidos para o RAM-disk, e este faz o papel de volume raiz

- Essa estratégia tem várias vantagens
- Transferências de dados entre partes da memória primária são operações muito rápidas, acelerando dramaticamente a carga do programa, o processamento de arquivos e o swapping
- O RAM-disk está sempre pronto para transferir os dados, sem precisar dos atrasos eletromecânicos de ativação inicial, e de posicionamento físico da mídia
- O RAM-disk não pode nunca ser removido. Isso nos impede de tantos irritantes erros de operação

- O primeiro permite mesmo que computador sem disco mas com um loop de fita o evento possa ser utilizado para executar o sistema P
- Tudo que for necessário ao programa pode ser carregado na ocasião do boot
- Somente quando o usuário terminar o trabalho e estiver pronto para sair do sistema os dados presentes no RAM-disk precisam ser salvos permanentemente em um positivo externo, por tratar-se de mídia volátil

- Um problema com esse esquema é que a mídia de memória primária no qual ele é implementado usualmente é volátil
- A restrição de endereçamento também se aplica a discos: um volume limita-se a 32767 blocos ou seja 16 megabytes, o que podia ser uma limitação para os discos rígidos da época: para usar discos maiores, pode ser necessário um particionamento adicional nos discos

9. Componentes do sistema P

- Uma unidade é um módulo de Pascal UCSD compilado separadamente, para futura inclusão no programa
- As unidades podem ser guardadas em arquivos de biblioteca para torná-los disponíveis em geral
- As unidades consistem de uma interface e de um corpo

- A primeira inclui declarações que descrevem as estruturas de dados, os cabeçalhos de procedimento, que descrevem as operações neles realizadas
- O corpo contém o código que implementa essas estruturas de dados e operações descritos no cabeçalho, e é mantido invisível ao usuário
- As unidades são gerenciadas pelo sistema P como se fossem segmentos
- A organização dos arquivos de código, discutidas anteriormente, e também a criação de bibliotecas de unidades de uso comunitário permite ao usuário usar partes do sistema operacional nos programas.

- É possível utilizar um utilitário de gerenciamento de bibliotecas para construir arquivos de código com unidades já incorporadas que tenham sido separadamente compilados. E assim que o sistema operacional foi construído
- A unidade chamada kernel contém código permanentemente residente o qual mantém o Code Pool, trata situações de erro, segmentos e faz leitura de segmentos que pode sofrer swapping para receber comandos, carregar programas inicializar o sistema e imprimir erros

- A forma como o sistema P inicia a execução de um programa de usuário é outro ponto a ser tratado aqui.
- Quando um programa deve ser executado seu segmento principal é mapeado como uma unidade chamada USERPROG, constrói-se o registro do ambiente de tempo de execução e o sistema ativa o programa do usuário como se fosse um dos seus próprios segmentos
- Algumas das unidades do sistema operacional, tais como KERNEL e SCREENOPS são acessíveis ao programador
- Outros como esses, que implementam o sistema de arquivos, podem ser usados somente através de unidades específicas da biblioteca do sistema

10. Independência do dispositivo

- A decisão de usar um monitor de vídeo como terminal para os usuários foi na época notável e bastante adequada.
- O problema de fazer o sistema ser baseado em um terminal de vídeo que fosse independente do dispositivo é algo tão complexo como o de fazer um sistema operacional ser portátil em relação a um conjunto de máquinas hospedeiras.
- De fato, para solucionar essa situação complexa foi necessária a elaboração da especificação de um pseudo terminal de vídeo que apresentasse as características de generalidade apropriadas.

- É preciso considerar quais códigos um usuário deve fornecer como entrada ao operar o sistema.
- Esse usuário pode querer parar uma tela de saída de dados que esteja rolando depressa demais para permitir-se ler.
- Mais tarde, poderá querer continuar a rolar a tela, ou então descartar algumas saídas, ou ainda abortar o programa em atividade naquele momento, através do acionamento de alguma tecla convencionada para provocar tal interrupção.
- Mesmo as mais óbvias teclas que forem adotadas para apagar um caractere ou uma linha variam de um terminal de vídeo para outro.

- Ao usar editores de tela padrão, o usuário pode querer pressionar teclas para mover o cursor para cima, para baixo, para a direita ou para a esquerda.
- Os códigos gerados por essas teclas podem não ser os códigos que deveriam ser enviados para a tela para mover o cursor para cima, para baixo, para a direita ou para a esquerda para que o cursor seja movido no sentido mencionado.
- O editor, e outros programas também, podem exigir que lhes sejam enviados códigos que não sejam confundidos com os associados às teclas usualmente empregadas para codificar caracteres de um texto usual.

- Usualmente são convencionadas sequências de escape formadas de diversos caracteres para codificar teclas especiais, por exemplo, tecla ESC seguida de uma letra, ou barra reversa seguida de um símbolo, etc.
- Esse problema se agrava quando se consideram editores baseados em tela e aplicações com uma interface de usuário orientada para tela - por exemplo, um programa de preenchimento de formulários.
- Esses exigem um método para especificação de como obter um endereçamento arbitrário de cursor, e quais códigos precisam ser enviados para fazer o cursor posicionar-se na coluna X linha Y da tela.

- Dois mecanismos foram usados para resolver esses problemas
- Um deles usa o arquivo chamado SYSTEM.MISCINFO cujo conteúdo é um conjunto de informações a respeito dos códigos enviados a partir do teclado e dos códigoa a serem enviados para a tela para que sejam executadas corretamente as diversas funções.
- O arquivo também contém informações por exemplo acerca da forma como deve ser o Code Pool externo.
- Caso já esteja de acordo com a convenção, informa também o local da memória principal no qual ele deve ser colocado

- Quando o sistema é iniciado, os valores em SYSTEM.MISCINFO são lidos, e aqueles que forem necessários quando o sistema estiver em execução são armazenados numa área de dados global do sistema localizado em alguma região do heap.
- Esses valores podem ser acessados pelo usuário através da unidade de sistema KERNEL
- O problema de endereçamento do cursor da tela foi resolvido solicitando ao fornecedor do sistema ou ao usuário que forneça um procedimento Pascal apropriado, através de uma unidade GOTOXY.

- Uma vez compilada, essa unidade pode ser colocada no arquivo de código do sistema operacional, para que possa ser utilizada tanto pelo usuário como pelo sistema.
- Todos os recursos para manuseio da tela do sistema, incluindo apresentação do prompt, são disponibilizados ao programador através de uma unidade chamada SCREENOPS

11. Uso do p-system em rede

- Liaison (= ligação) é o nome de uma extensão do sistema p que permite que computadores possam ser conectados (ligados) entre si formando uma rede local
- Suas metas primárias consistem em prover meios a serem usados para que programas portáteis distribuídos sejam escritos.
- Através dele, softwares existentes do p-system podem se comunicar e compartilhar recursos de hardware ou software
- Arquivos muito grandes em um disco rígido, e processadores rápidos de ponto flutuante são exemplos de hardwares que podem funcionar como recursos adicionais desejáveis.
- Bancos de dados e um sistema de gerenciamento de banco de dados são bons exemplos de recursos de software compartilháveis

- Da mesma forma como o sistema P abstrai arquiteturas de máquinas em uma pseudo máquina, Liaison abstrai arquiteturas de rede em uma pseudo rede.
- Essa abstração foi implementada em arquiteturas de rede reais tais, tais como omninet e ethernet através da construção de um BIOs estendido para o sistema P
- Os recursos de rede do Liaison são obtidos adicionando-se unidades apropriadas de rede como componentes do sistema P
- Essas unidades fornecem funções de alto nível tais como a transmissão e recepção especificamente endereçadas ou então mensagens de broadcast enquanto drivers de rede adicionados ao Bios fornecem interface para o Hardware da rede

- Liaison é baseado no modelo de rede cliente-servidor
- Nesse paradigma um programa em p-code (ou processo) pode prover algum serviço do qual outros programas na rede possam se utilizar.
- Dessa forma um serviço é produzido por um servidor e é utilizado por um cliente
- Um servidor pode disponibilizar seus serviços para vários clientes simultaneamente
- Um cliente pode fazer uso de diversos serviços ao mesmo tempo.
- A figura 9.16 mostra um exemplo de uma rede Liaison.

- Cada um dos processadores nessa figura pode ser diferente dos demais
- Conexões físicas são mostradas em linhas sólidas e conexões lógicas são mostradas em linhas tracejadas
- O diagrama mostra diversos computadores baseados em diferentes processadores, que constituem os nós da rede
- Por serem eles frequentemente usados como sistemas computacionais de propósito geral, esses nós são usualmente chamados estações de trabalho

- Um computador E possui um disco de alta capacidade
- Outro, F, tem uma impressora de alta velocidade e também um disco de grande capacidade.
- Todos os outros nós dispõem apenas de discos flexíveis e consoles para o operador, sem outros periféricos
- O computador E pode ser usado como um servidor de disco, enquanto o computador F, como um servidor de disco ou então como um servidor de impressora
- As demais estações de trabalho podem usar os serviços disponibilizados por esses dois servidores ou então comunicar-se diretamente com cada um dos outros enquanto executam algum programa de aplicação distribuída

- Liaison dispõe de localização dinâmica de servidores e clientes: eles podem identificar um ao outro antes de sua interação se completar
- Essa flexibilidade permite que serviços sejam deslocados para diferentes estações de trabalho se for necessário
- Um dos mais importantes recursos de Liaison é sua facilidade de permitir a existência de software não distribuído que faça o uso dos serviços da rede

- Esse recurso é oferecido botando o sistema com vários servidores e programas acessar esses servidores
- Existem três servidores padrão: o servidor de disco, o servidor de semáforo, e o servidor de impressora
- Através de um utilitário menu-driven SHARE, clientes podem utilizar os serviços por eles oferecidos requisitando-os por intermédio de um nível de menu para cada tipo de servidor

Servidores de disco

- O servidor de disco é um programa que permite o acesso a discos locais ao servidor a qualquer programa cliente dos usuários Liaison na rede.
- Qualquer volume do sistema P ao qual o servidor tem acesso pode ser disponibilizado aos clientes desde que o volume não esteja sendo partilhado com algum outro servidor
- Em outras palavras se dois ou mais servidores tiverem acesso a um volume de dados, apenas um deles pode oferecê-lo como serviço, a qualquer momento

- Quando o servidor de disco é executado numa estação de trabalho, o operador (usualmente o gerente da rede específica) especifica qual dos volumes locais da estação de trabalho deve ser disponibilizado para a rede.
- Esses volumes são então disponibilizados aqui a qualquer cliente, enquanto o servidor estiver sendo executado.
- O servidor não pode terminar sua execução enquanto existirem clientes utilizando-o

- Para usar o servidor de disco, é preciso executar o programa SHARE para estabelecer conexão entre o sistema P do cliente e o do servidor.
- O usuário pode obter uma lista de todos os volumes disponíveis em algum dos servidores de disco presentes na rede

- Ele pode também ganhar acesso a um ou mais deles montando volumes remotos em lugar de qualquer dos volumes locais principais (sub-volumes não podem ser substituídos).
- Essa montagem é uma operação lógica, e faz com que os volumes remotos sejam conhecidos do sistema P
- Os volumes devem estar fisicamente online no servidor para que este possa disponibilizá-los
- Uma vez que o usuário tenha montado localmente o volume remoto, ele poderá passar a utilizá-lo como se fosse um volume local
- Dessa forma usuário pode utilizar o sistema de arquivos para manipular o volume remoto ou executar um programa para processar dados sobre ele

- Montar um volume remoto para uso local torna-se algo parecido com substituir um disco flexível.
- Nenhum usuário pode estender ou apagar um arquivo que esteja aberto

Servidores de semáforo

- O servidor de semáforo permite que os clientes indiquem que desejam fazer uso exclusivo de um recurso (alocação dedicada)
- Isso é muito utilizado quando mais de um usuário da rede puder acessar um mesmo arquivo específico
- Se esses usuários, antes da execução do seu programa, acionarem o programa SHARE, eles poderão informar-se se o arquivo desejado está ou não disponível naquele momento

- O usuário que em primeiro lugar esperar pelo semáforo que foi associado ao recurso deverá obter acesso a ele
- Aos usuários subsequentes em espera por ele, será negado o acesso
- Como o servidor de semáforo na maior parte das vezes é utilizado para controlar o acesso a um arquivo, ele é executado como um processo concorrente com esse particular servidor de disco

- Pode haver apenas um servidor de semáforo de cada vez para evitar que dois clientes utilizem um mesmo semáforo em servidores diferentes
- A maior fraqueza do servidor de semáforo é sua natureza voluntária
- Todos os usuários que queiram partilhar o acesso a um recurso devem concordar em usar esse servidor e um mesmo protocolo, caso contrário, o controle do acesso irá falhar

Servidor de impressora

- O servidor de impressora é parecido com o servidor de disco, mas seu propósito é providenciar serviços SPOOLed de saída aos dispositivos seriais
- Trata-se de um programa separado do servidor de disco e não pode ser executado simultaneamente em uma estação de trabalho

- O servidor de impressora utiliza seu disco local para efetuar SPOOLing das saídas dos clientes para dispositivos do tipo impressora
- O cliente pode selecionar o dispositivo em oferta por um servidor de impressora, e montar este dispositivo no lugar de um dos seus próprios volumes seriais locais

- Quaisquer dados de saída de um cliente direcionados àquele dispositivo será SPOOLed para um servidor local de disco, até que um comando Break seja enviado ao servidor
- Força o servidor a imprimir o arquivo assim que alguma impressora acessível a ele se torne disponível
- Um comando Break pode ser imposto explicitamente por um programa, mas ele é também executado normalmente pelo programa que criou a saída, quando este chega ao seu final

Programação em Rede

- Liaison pode ser utilizado por programadores em diversos níveis:
  - Ele pode ser ignorado: isso não significa que o usuário de um programa não possa utilizar a rede, mas que o programador não precisa estar atento a quais serviços estejam sendo oferecidos, localmente ou na rede
  - O programador pode fazer o uso de unidades do sistema operacional Liaison. Isso para ver interfaces de alto nível a rede compartilhar disco impressora e semáforos

- Essas unidades oferecem meios de interrogar servidores remotos solicitando que sejam informados os serviços disponíveis (o servidor de disco poderia retornar o nome do servidor e os nomes dos volumes que estão sendo disponibilizados) bem como os meios de estabelecer tais serviços
- Unidades também são disponíveis que implementam requisitos do cliente enquanto este estiver usando o servidor de impressora (tais como forçar a impressão de um documento antes que o programa do cliente termine)
- O programador pode portanto programar o uso dos serviços de rede de forma totalmente transparente ao usuário.

- O programador pode implementar serviços de rede escrevendo um programa de servidor e um programa ou unidade de cliente de tal maneira que um novo serviço seja oferecido em geral para a rede
- Exemplo: um servidor de banco de dados poderia implementar com acesso a banco de dados usando um único gerenciador de banco de dados enfileirando requisições provenientes de qualquer usuário
- Uma vantagem desse tipo de solução é que pode proporcionar substancial redução no uso da rede
- Os próprios serviços de rede podem ser escritos em uma série de níveis. Esses níveis são disponibilizados explicitamente por Liaison para serem usados na comunicação entre usuários

- Esses níveis são os seguintes:
  - Rotinas de baixo nível para transmissão e recepção na rede – essas rotinas tentam enviar ou receber mensagens, mas não garantem sucesso. É o nível mais baixo (socket level)
  - Comunicação confiável usuário a usuário – essas rotinas têm a responsabilidade de assegurar ao remetente que as mensagens de fato chegaram a seu destino. São disponibilizadas rotinas para estabelecer permitir o uso e finalmente desconectar uma uma ligação de usuário para usuário (é chamado nível de canal).
  - Remote server location – essas rotinas ajudam a determinar a existência de um servidor remoto e estabelecer comunicações com ele

- Liaison fornece um ambiente de desenvolvimento de software de rede
- O programador ou usuário não precisa se preocupar em relação a qual hardware deve ser usado – isso fica a cargo do sistema operacional, portanto resta ao usuário apenas determinar se o sistema está ou não implantado naquela rede real

- Um exemplo de quão transparente pode ser para o usuário o software escrito para ser nele executado:
- O proprietário do microcomputador deseja imprimir uma carta
- O proprietário executa um pequeno programa de menu no sistema de D e seleciona a opção correspondente, de processamento de palavras

- O programa de menu aciona o computador remoto B, que contém o programa de processamento de palavras e o dispositivo de armazenamento de dados
- Ele monta automaticamente volumes apropriados, e imediatamente executa o software
- Quando o proprietário de D termina o programa de processamento de palavras, retorna ao programa de menu, e libera então a informação de todos os volumes necessários para o processamento de palavras

- O programa de menu em D espera então pela próxima seleção de um comando do menu
- O proprietário não precisa ter consciência de que o software está sendo executado na rede
- Se B não estiver disponível por qualquer razão então o proprietário de D é informado de que o servidor não está disponível, retornando ao programa de menu

- Foi dito anteriormente que software pode ser considerado como um recurso escasso
- Como o software de microcomputadores costuma ser cobrado por cópia, uma organização pode resolver comprar somente uma cópia do programa e disponibilizá-lo no hardware de rede
- O desenvolvedor do software por outro lado vai querer ser pago pelas n cópias do programa que foram disponibilizadas para as n estações de trabalho individuais na rede

- Um recurso (Liaison monitor) para controlar o uso de programas é disponibilizado no sistema operacional Liaison
- Esse recurso permite que um desenvolvedor de software ou gerente de rede evite que o mesmo software seja multiplicado em uma rede mais que para um número fixo conhecido de pessoas
- Por exemplo o limite de 3 usuários estabelecido para o programa de gerenciamento de banco de dados. Se um quarto usuário tentar executar o programa quando três já estejam usando ele será informado de que o limite da rede foi excedido e que alguns usuários deverão terminar uso do programa antes que esse novo usuário possa acessá-lo

- O limite para a rede pode ser estabelecido para um programa consultando-se um utilitário destinado a este propósito
- O advento da versão Liaison do sistema P foi um passo adiante no desenvolvimento do sistema
- Liaison foi um dos produtos pioneiros desenvolvidos para rede, independentes de Hardware