**Tanenbaum (Micro Kernel) x Torvalds (Kernel Monolítico)**

Um kernel monolítico é um kernel em que todos os serviços (sistema de arquivos, VFS, drivers de dispositivo etc.), bem como a funcionalidade principal (agendamento, alocação de memória, etc.) são um grupo unido que compartilha o mesmo espaço. Isso se opõe diretamente a um microkernel.

Um microkernel prefere uma abordagem em que a funcionalidade principal é isolada dos serviços do sistema e dos drivers de dispositivo (que são basicamente apenas serviços do sistema). Por exemplo, o VFS (sistema de arquivos virtual) e os sistemas de arquivos do dispositivo de bloco (ou seja, minixfs) são processos separados que são executados fora do espaço do kernel, usando o IPC para se comunicar com o kernel, outros serviços e processos do usuário. Em resumo, se é um módulo no Linux, é um serviço em um microkernel, indicando um processo isolado.

Não devemos confundir o termo kernel modular como algo menos monolítico. Alguns kernels monolíticos podem ser compilados para serem modulares (por exemplo, Linux), o que importa é que o módulo seja inserido e executado no mesmo espaço que lida com a funcionalidade principal (espaço do kernel).

A vantagem para um microkernel é que qualquer serviço com falha pode ser facilmente reiniciado, por exemplo, não há interrupção do kernel se o sistema de arquivos raiz lançar um aborto. Isso também pode ser visto como uma desvantagem, pois pode ocultar bugs bastante críticos (ou fazê-los parecer não tão críticos, porque o problema parece se corrigir continuamente). É visto como uma grande vantagem em cenários em que você simplesmente não pode consertar algo convenientemente depois que ele foi implantado.

Do ponto de vista de Linus, os micro-kernels requerem algoritmos distribuídos que são desagradáveis pela falta de uma referência de tempo comum, juntamente com possíveis mensagens perdidas e a incerteza sobre o processo remoto estar parado ou apenas lento. Nenhum desses problemas se aplica a sistemas operacionais baseados em microkernel em uma única máquina.

Além disso, a maioria dos componentes do espaço do usuário são drivers e possuem interações muito diretas com os servidores. O número de servidores de espaço do usuário é bastante pequeno: um servidor de arquivos, um servidor de processos, um servidor de rede, um servidor de reencarnação e um repositório de dados e muito mais. Cada um tem um trabalho bem definido e uma interação bem definida com o resto do sistema.

Para Linus deve-e evitar ao máximo estruturas de dados compartilhadas. Os sistemas devem ser compostos de módulos pequenos que ocultam completamente suas estruturas de dados internas de todos os outros. Eles devem ter interfaces 'finas' bem definidas que outros módulos podem chamar para realizar o trabalho. Esse é o trabalho da programação orientada a objetos - ocultar informações - não compartilhá-las.