5
Conclusão

Nesta dissertação, apresentamos os conceitos de sistema operacional scriptável e de *scripting* de *kernel*, os quais são resultados da aplicação das idéias de extensibilidade através de *scripting* e de desenvolvimento de programas utilizando linguagens de script aos conceitos de sistema operacional e sistema operacional extensível.

Nós implementamos Lunatik, a nossa infra-estrutura que provê suporte para scripting de kernel de sistema operacional com Lua, permitindo o desenvolvimento e extensão de *kernels* usando scripts Lua. Usando Lunatik, desenvolvedores de kernel podem tornar subsistemas "scriptáveis" e usuários podem carregar e executar scripts dinamicamente em um interpretador embutido no *kernel*. Lunatik é uma prova de conceito de que é possível prover um ambiente para o *scripting* de *kernel* similar àqueles para aplicações em espaço de usuário.

Nós acreditamos que nossa abordagem para *scripting* de *kernel* pode auxiliar na mitigação dos problemas conhecidos de sistemas operacionais: flexibilidade, confiabilidade, desempenho e facilidade de desenvolvimento.

Acreditamos também que essa abordagem para *scripting* de *kernel* pode auxiliar na inovação em desenvolvimento de *kernel* de sistema operacional. Primeiro, ela disponibiliza um ambiente de programação interpretado e de mais alto nível para o *kernel*. Portanto, ela torna a prototipação e experimentação mais rápidas. Segundo, ela disponibiliza uma plataforma de extensibilidade mais fácil e ágil do que módulos de kernel e mais flexível do que ajuste de parâmetros. Portanto, em subsistemas previamente adaptados para serem scriptados, ela permite que não-desenvolvedores de *kernel*, como desenvolvedores de aplicações e administradores de sistemas, possam fazer experimentos com o *kernel* para atender as suas próprias necessidades.

A implementação de Lunatik para NetBSD foi desenvolvida, em parte, dentro do programa *Google Summer of Code*. Essa implementação está acessível em `http://netbsd-soc.sourceforge.net/projects/luakern/`, junto com um relatório sobre o desenvolvimento, um pequeno tutorial e um manual de uso da interface de programação de *kernel*. Atualmente, Lunatik está

PUC-Rio - Certificação Digital Nº 0912839/CA

sendo desenvolvido no escopo do projeto NetBSD para possivelmente ser incorporado nesse sistema operacional, no futuro[1]. A implementação de Lunatik para Linux está acessível em `http://sourceforge.net/projects/lunatik/`.

A seguir discutiremos algumas lições sobre *scripting* de *kernel* que aprendemos durante o desenvolvimento desta dissertação.

5.1 Lições Aprendidas

Ao longo do desenvolvimento das implementações de Lunatik para Linux e NetBSD, pudemos aprender algumas lições sobre a implantação de *scripting* de *kernel*:

- A principal lição que tivemos é que adaptar um *kernel* para ser scriptado é muito mais complexo do que simplesmente embutir o interpretador de uma linguagem de script nele. Isso decorre, sobretudo, do fato de que a tarefa de desenvolver e estender *kernels* em si é bastante complexa.
- Existem diferentes contextos de execução em um *kernel* e isso impacta diretamente no desenvolvimento do interpretador da linguagem de script. Por exemplo, são necessários diferentes alocadores de memória, dependendo do contexto de execução.
- A cópia de grandes quantidades memória pode ser bastante custosa. Por isso, é importante aproveitar as implementações internas de estruturas de dados do *kernel* sempre que possível. Por exemplo, ao implementarmos o exemplo de filtragem de dados em *kernel* do capítulo 4 (função `filter` da figura 4.1), obtivemos uma grande perda de desempenho ao copiarmos dados criando novas *strings* em Lua (usando a função `lua_pushstring` da API de Lua). Nesse caso, o mais indicado seria mapear os dados (e.g., em um *user datum*) para serem acessados em Lua.
- A criação de uma infra-estrutura de *scripting* de *kernel* também é mais complexa do que simplesmente embutir um interpretador no *kernel*. São necessários mecanismos para a carga de scripts e formas de integrar a linguagem de script com o *kernel*.
- O *scripting* de *kernel* se posiciona entre o ajuste de parâmetros em tempo de execução (e.g., Sysctl e Sysfs) e a criação de módulos de *kernel*. É importante buscar aplicações que caibam nesse nicho, utilizando a ferramente correta (ajuste de parâmetros, módulos de *kernel* ou *scripting* de *kernel*) para cada necessidade.

[1]Anúncio pode ser visualizado em `http://netbsd.org/changes/index.html#newdev201101`

PUC-Rio - Certificação Digital Nº 0912839/CA