Sistemas Distribuídos

Relatório do Primeiro Trabalho Prático

Universidade Federal do Rio de Janeiro

Aluno: Andrew Cruz

DRE: 114136204

Professor: Daniel Ratton

Introdução

Este trabalho consiste na implementação de técnicas de IPCs (Inter Process Communication) baseadas em trocas de mensagens entre os processos. Este tipo de comunicação é primitivo no contexto de sistemas operacionais, dependendo de System Calls e da utilização do Kernel space.

Envio e Recebimento de Signals

Aqui, a comunicação é realizada por meio de uma função que tem como parâmetros o id do processo que está enviando o sinal e o id do processo que está recebendo. Estes identificadores são únicos para cada processo enquanto ele estiver rodando (estado running). A função *kill* recebe como argumentos o número que representa o tipo de sinal e o id do processo alvo que foi passado para a função *main*. Se *kill* retornar –1, o processo não existe, fazendo com que uma mensagem de erro seja retornada. Do contrário, o sinal é enviado e o programa retorna uma mensagem de sucesso.

Os sinais também possuem identificadores, de acordo com a tabela abaixo:

1. SIGHUP	2. SIGINT	3. SIGQUIT	4. SIGILL
5. SIGTRAP	6. SIGABRT	7. SIGEMT	8. SIGFPE
9. SIGKILL	10. SIGBUS	11. SIGSEGV	12. SIGSYS
13. SIGPIPE	14. SIGALRM	15. SIGTERM	16. SIGURG
17. SIGSTOP	18. SIGTSTP	19. SIGCONT	20. SIGCHLD
21. SIGTTIN	22. SIGTTOU	23. SIGIO	24. SIGXCPU
25. SIGXFSZ	26. SIGVTALRM	27. SIGPROF	28. SIGWINCH
29. SIGINFO	30. SIGUSR1	31. SIGUSR2	

Manipulação de Sinais

Como na especificação, o programa recebe como parâmetro o tipo de espera do sinal, sendo *busing wait ou blocking wait*. Foram adotados três sinais: SIGSTOP(17), SIGCONT(19) e SIGHUP(1). A função *manSinal* trata entrada, sendo que, quando um sinal do tipo SIGHUP é recebido, o programa é encerrado.

Como visto na teoria, quando um processo está em modo de *busy wait,* ele está basicamente num loop infinito, rodando até que receba um sinal.

Já em blocking waiting, o sistema executa um *pause* iterativamente, passando a ficar em *waiting*, ou seja, liberando o CPU para trabalhar com outros processos. Quando outro sinal for recebido, ele retorna à execução.

Endereço do Repositório

https://github.com/andrewcruz16/SD2018.1

Referências

https://www.cyberciti.biz/faq/unix-kill-command-examples/

http://www.yolinux.com/TUTORIALS/C++Signals.html

https://en.wikipedia.org/wiki/Signal_(IPC)

http://stackoverflow.com