

Publicidade



ASSINE 0800 703 3000

BATE-PAPO

E-MAIL

SAC

Voip

E-Mail Grátis

Shopping

ÍNDICE PRINCIPAL

PROCURAR:

no site

OK

Terça, 24/02/2009

- » Introdução
- » Programação
- » Administração
- » Hardware
- » Aplicativos
- » Jogos
- » Segurança
- » Editorial
- » Entrevistas

ARTIGOS

- » Fórum
- » Links
- » Notícias
- » Pegue o Linux
- » Documentação

COMUNIDADE

- » Programas
- » Dúvidas
- » Oportunidades
- » Sobre
- » Contato
- » Publicidade

SERVIÇOS

Powered By:
DEBIAN
GNU/LINUX

English Version

Linux Solutions

Shopping
OLinux

Programação

Tutorial de sockets - Parte VII

Por: [Frederico Perim](#)

Quando você estiver enviando estes dados, você pode estar seguro em usar uma função similar a `enviatudo()`, acima, assim você sabe que todos os dados são enviados, mesmo que leve múltiplas chamadas `send()` para completar o processo.

Da mesma forma, quando estiver recebendo dados, você precisa fazer um pouco de trabalho extra. Para ter certeza, você deve assumir que recebeu parte dos dados. Assim precisamos chamar `recv()` várias vezes até que o pacote completo tenha chegado.

Mas como? Bem, nós sabemos o número de bytes que precisamos receber ao todo para que o pacote esteja completo, já que este número é anexado na frente do pacote. Nós também sabemos que o tamanho máximo do pacote é: $1+8+128$, ou 137 bytes (porque foi assim que definimos o pacote.)

O que você pode fazer é declarar um array grande o suficiente para dois pacotes. Este é a sua array de trabalho onde você irá reconstruir pacotes assim que chegarem.

Toda vez que receber (`recv()`) dados, você os passa para a array e checa se o pacote está completo. Ou seja, o número de bytes no buffer (array) é maior ou igual em relação ao número especificado no cabeçalho (+1, porque o tamanho no cabeçalho não o inclui o byte para o tamanho em si). Se o número de bytes no buffer é menor que 1, o pacote não está completo, obviamente. Você vai ter que fazer algo especial em relação a isso, já que o primeiro byte é lixo e não pode ser confiável para se averiguar o tamanho do pacote.

Uma vez que pacote esteja completo, você pode fazer o que quiser. Use e remova de seu buffer.

Nossa você já absorveu isso na sua mente? Bem, então aí vai a segunda parte: você pode ter lido todo o primeiro pacote e parte do segundo com uma única chamada `recv()`. Ou seja, você tem um buffer com um pacote completo, e parte do próximo pacote! Que saco (Mas isso é porque você fez um buffer grande o suficiente para suportar dois pacotes!)

Já que você sabe o tamanho do primeiro pacote através de seu cabeçalho, e andou checando o número de bytes no buffer, você pode subtrair e calcular quantos bytes pertencem ao segundo (incompleto) pacote. Quando você já cuidou do primeiro, pode limpá-lo do buffer e mover parte do segundo para frente do buffer para que esteja pronto para próxima chamada `recv()`.

Bem, realmente não é algo trivial. Agora você precisa praticar e logo tudo isso fará sentido naturalmente. Eu juro!!!!

[< Anterior](#)

ENQUETE

Com qual frequência você
acessa o site Olinux?

- ☐ Todos os dias
- ☐ Uma vez por semana
- ☐ Cinco vezes aos mês
- ☐ Poucas vezes ao mês
- ☐ Outra


VOTAR

NEWSLETTER

Inscreva-se e receba as últimas
notícias, programas, artigos,
novidades e tudo do mundo
Linux que aconteceu na semana.

Digite seu email:

OK

 **Lidando com `send()` parciais** **Filho do Empacotamento de Dados**

Enviar para um amigo



Imprimir Índice de artigos



Jaguar



Vários modelos.
Entre e confira.

Brinquedos



das Meninas Super
Poderosas. Clique!

Filmadora



Multilaser CR-518
Digital.
Compare!

Esteira



Entre em forma
antes do verão.

COMPARE PREÇOS

Buscar

[Publicidade](#) / [Sobre OLinux](#) / [Entre em Contato](#) / [Privacidade](#)
Copyright (c) 2000-2007, OLinux - O Portal de Linux do Brasil.
Desenvolvido por: [Linux Solutions](#)
Todos os Direitos Reservados.