**长沙理工大学**

《网络协议编程课程设计》报告

**telnet口令嗅探工具**

徐永林



**学 院 计算机与通信工程 专 业 网络工程**

**班 级 网络15-2 学 号 201558080231**

**学生姓名 徐永林 指导教师 夏卓群**

**课程成绩 完成日期 2018年 1月 5日**

**网络协议编程课程设计成绩评定**

**院 系 计算机与通信工程 专 业 网络工程**

**班 级 网络1502 学 号 201558080231**

**学生姓名 徐永林 指导教师 夏卓群**

**指导教师对学生在综合实训中的评价**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评分项目 | 优 | 良 | 中 | 及格 | 不及格 |
| 学习态度与遵守纪律情况 |  |  |  |  |  |
| 课程设计完成情况 |  |  |  |  |  |
| 课程设计报告的质量 |  |  |  |  |  |

指导教师成绩 指导教师签字 年 月 日

**课程设计答辩组对学生在综合实训中的评价**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评分项目 | 优 | 良 | 中 | 及格 | 不及格 |
| 课程设计完成情况 |  |  |  |  |  |
| 课程设计报告的质量 |  |  |  |  |  |
| 课程设计答辩 |  |  |  |  |  |

答辩组成绩 答辩组长签字 年 月 日

**课程设计综合成绩**

注：课程设计综合成绩＝指导教师成绩×60%＋答辩组成绩×40%

**课程设计任务书**

计算机与通信工程学院 网络工程专业

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 课程名称 | 网络协议编程课程设计 | 时间 | | 2017～2018学年第1学期 17～18周 | |
| 学生姓名 | 徐永林 | | 指导老师 | | 夏卓群 |
| 题 目 | Telnet口令嗅探工具 | | | | |
| 主要内容：  (1) 用户计算机连接远程计算机，输入远程计算机的用户名和密码，便可加入聊天室。  (2) 在终端上输入的数据，在后台都会有过滤，解析，拼装成原文的过程。  (3) 用户第一次输入的数据，会作为用户名，后面输入的数据会作为聊天的内容 | | | | | |
| 要求：  (1) 结合网络协议编程课程设计所学知识设计并实现telnet数据包捕获。  (2) 学会文献检索的基本方法和综合运用文献的能力。  (3) 通过课程设计培养严谨的科学态度，认真的工作作风和团队协作精神。 | | | | | |
| 应当提交的文件：  (1) 课程设计学年论文。  (2) 课程设计附件（相关图纸、设备配置清单、报告等）。 | | | | | |

**Telnet口令嗅探工具**

**学生姓名：徐永林 指导老师：夏卓群**

摘 要 网络嗅探技术是一种常见的的网络数据采集和监听方式。本次课程设计是基于node.js利用net模块监听TCP层，采集数据包的原理，在装有linux子系统的win10平台下，采用net模块来设计并实现的一个聊天室程序，实现了对用户进行Telnet远程登陆后输入的口令的捕获，并扩展了聊天的功能。经测试，实验效果达到预期的效果。

关键词 网络嗅探；net；Telnet；聊天室

**Telnet password sniffer**

Student Name: Xu Yonglin Advisor：XIA Zhuo-qun

**Abstract**  Network sniffing is a common method of network data acquisition and monitoring. This course is designed by using the net node.js module monitors the TCP layer based on the principle of data collection package, equipped with Linux system under the win10 platform, a chat program using net module to design and implement, capture the Telnet remote login to the user input password, and extended chat function. The experimental results have achieved the expected effect.

**Key words** network sniffing；net；Telnet； chat room

**1 引言**

此课程设计是基于telnet协议用node语言开发的嗅探器，并由此扩展为聊天室，使用webstorm平台开发程序并测试。

**1.1选题的背景**

繁忙的工作使人们没有时间去跟远方的亲戚朋友相聚,昂贵的电话费使我们不能聊的尽兴，因此网络聊天诞生了，并且成为了时下极为流行的一种沟通方式,特别是在现代高新的技术推动下,在线聊天呈现出良好的势头和广阔的前景。此次设计旨在开发出一个高效的网上聊天室，并作为毕业设计研究课题具有非常重要的研究价值。

**1.2 选题的目的和意义**

这次设计的课题正是互联网与社会生活之间的紧密关系的体现，现今的社会，人们已经离不开了网络，网络已经成为人与人之间交流的一种形式，他能够把事情的复杂化转为简单化。伴随着互联网的飞速发展，对作为人们交流的研究和设计变得越来越重要。同时人们更亲睐可以实时进行沟通的方式，于是在线聊天室应运而生。本程序通过telnet嗅探捕获数据包，实现聊天室的功能。

**1.3设计平台**

在带有linux子系统的Windows10的平台下，以Webstorm为开发环境，使用node语言编写程序。并使用Windows10自带的DOS平台进行测试。

**2设计原理**

**2.1 node语言与net模块**

Node.js是一个Javascript运行环境，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装。Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好[1]。

V8引擎执行Javascript的速度非常快，性能非常好。Node.js是一个基于Chrome JavaScript运行时建立的平台，用于方便地搭建响应速度快、易于扩展的网络应用。Node.js使用事件驱动，非阻塞I/O模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用[2]。

Net模块是node.JS针对TCP的一个模块，TCP服务在网络应用中十分常见，目前大多数的应用都是基于TCP搭建而成的。net模块提供了一个异步网络包装器，用于TCP网络编程，它包含了创建服务器和客户端的方法。

**2.2 Telnet协议原理**

Telnet协议是TCP/IP协议族中的一员，是Internet远程登陆服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。在终端使用者的电脑上使用telnet程序，用它连接到服务器。终端使用者可以在telnet程序中输入命令，这些命令会在服务器上运行，就像直接在服务器的控制台上输入一样。可以在本地就能控制服务器。要开始一个telnet会话，必须输入用户名和密码来登录服务器。Telnet是常用的远程控制Web服务器的方法。Telnet服务是采用一种客户/服务器结构，具体的工作原理如下：

当用户用Telnet登录进入远程计算机系统时，事实上启动了两个程序，一个叫Telnet客户程序，它运行在的本地机上；另一个叫Telnet服务器程序，它运行在要登录的远程计算机上。本地机上的客户程序要完成如下功能：

1、建立与服务器的TCP 连接；

2、从键盘上接收你输入的字符；

3、把输入的字符串变成标准格式并送给远程服务器；

4、从远程服务器接收输出的信息；

5、把该信息显示在屏幕上。

远程计算机的“服务”程序通常被称为“精灵”，它平时不声不响地候在远程计算机上，一接到你的请求，就马上活跃起来，并完成如下功能：

1、通知你的计算机，远程计算机已经准备好了；

2、等候输入命令；

3、对命令作出反应（如显示目录内容，或执行某个程序等）；

4、把执行命令的结果送回给计算机；

5、重新等候命令。

**2.3 网络嗅探原理**

网络嗅探是一种常用的收集网络数据包的方法,其基本原理对经过网卡的数据包进行捕获和解码,从链路层协议开始进行解码分析,一直到应用层的协议,最后获取数据包中需要的内容。

**2.4 网卡工作模式**

在实际的系统中,数据的收发由网卡来完成,网卡的主要工作原理是整理计算机发往网线上的数据,并将数据包发送出去。当网卡接收到传输来的数据时,根据接收数据帧的目的MAC地址和网卡驱动程序设置的接收模式进行判断,对需要接收的就产生中断信号送CPU,然后由操作系统调用动程序进行接收；认为不该接收的就丢弃不管,所以不该接收的数据在网卡处就截断了。对于网卡来说一般有四种接收模式:

（1）广播方式:能够接收网络中的广播信息。

（2）组播方式:网卡能够接收组播数据,也就是一个人发出的包可以同时被其他多个有资格的人接收,这些人就形成了一个组,他们在组内的通信是广播式的。

（3）直接方式:只有目的网卡才能接收该数据。

（4）混杂模式：能够接收一切通过网卡的数据。

**2.5数据包捕获机制**

网络数据包截获机制一般指通过截获整个网络的所有信息流量,根据信息源主机,目标主机,服务协议断口等信息简单过滤掉不关心的数据,再将用户感兴趣的数据发送给更高层的应用程序进行分析。在局域网中,由于以太网是基于广播方式传送数据的,以太网上的所有主机都共享一条网络总线,所以所有的物理信号都会被传送到每一个主机节点,如果将网卡设置为混杂接受模式,则无论监听到的数据帧目的地址如何,网卡都能予以接收。而Telnet协议是以明文形式在网络上传输,其包含了登陆时输入的用户名和密码口令。因此可以设计程序监听到所有局域网内的数据通信,得到这些敏感信息。

嗅探器作为一种网络通讯程序,主要是通过对网卡的编程来实现的,对网卡的编程有很多种方法,通常的套接字程序只能响应与自己硬件地址相匹配的或是以广播方式发出的数据帧,对于其他形式的数据帧,网络接口在验证目的地址并非自身地址之后将不引起响应,也就是说应用程序无法接收到达的数据包。而网络嗅探器的目的恰恰在于从网卡接收所有经过它的数据包,这些数据包既可以是发给自己的也可以是发给别人的。显然,要达到此目的就不能让网卡按通常的正常模式工作,而必须将其设置为混杂模式。嗅探程序的设计一般有如下几个阶段：

（1）包捕获设置：包括网络适配器的识别以及网卡混杂模式的设置。

（2）过滤器设置：对数据包的捕获设定一些条件,如可以按某些特定的IP地址进行过滤，也可根据网络协议只捕获某种特定的协议,如TCP或UDP包中的数据，（如该课程设计就根据端口只捕获Telnet协议）过滤出包含特定信息的数据包，例如包含用户名和用户发送的信息内容的数据包。

（3）数据分析：对捕获的数据进行分析处理，获得数据包中的上层信息。本程序功能主要是聊天，也就是数据的交互，因此会对包里的数据还原，并广播到聊天室内其他成员，从而达到通讯的目的。

本程序主要是通过node的模块net进行数据抓包，并解析用户发送的数据信息。

**3 设计步骤**

**3.1程序流程设计**

用户在带有linux子系统的win10系统中，打开DOS平台，输入相关命令，进入聊天系统界面，就可以进行聊天，流程设计如图3-1所示。

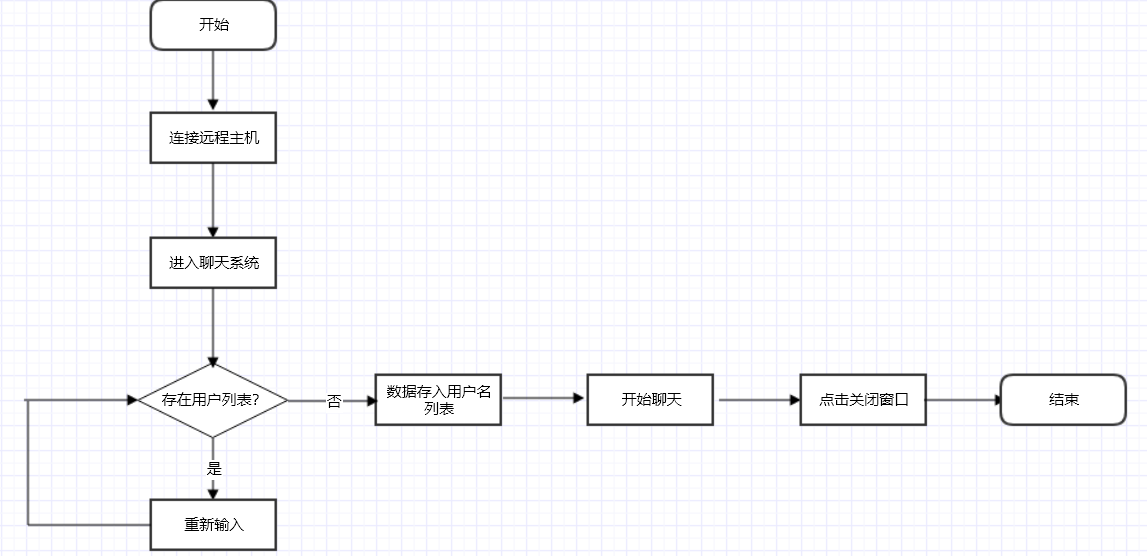


图3-1.程序流程图

**3.2主要函数模块说明**

引入net模块一般赋值给net对象，这是下面所有业务逻辑的基石，下面所有的函数都是这个对象的属性，主要函数如下：

1. createServer函数，该函数是对服务器的监听事件，当用户连接服务器时，会触发该函数，用户名的存储，信息的广播，都是在该函数里进行的，大致代码，及注释如下：

let server = net.createServer(function (socket) { //创建服务器，用户连接服务器时触发的函数

count++; //当用户连接服务器时，用户数量增加1

let nickname; //储存当前连接用户的用户名

socket.setEncoding('utf8'); //设置传递信息的编码

// 页面tip

socket.write(

'\n > welcome to \033[92mgroup-chat\033[39m!'

+ '\n > ' + count + ' people are connected at this time.'

+ '\n > please write your name and press enter: '

);)}

1. broadcast函数，该函数的功能主要是当用户发出消息时，系统将信息发送给其他用户，大致代码如下：

let broadcast = (msg, exceptMyself) => {

for (var i in users) {

if (!exceptMyself || i != nickname) {

users[i].write(msg);

}

} };

1. data函数，该函数是用户发送消息时触发的，主要用于数据的解析，过滤和包装以及业务逻辑处理。大致代码如下：

socket.on('data', function (chunk) {

socket.setEncoding('utf8');

// 删除回车符

chunk = chunk.replace('\r\n', '');

if (!nickname) {

if (users[chunk]) {

socket.write('\033[93m> nickname already in use. try again:\033[39m ');

return;

} else {

nickname = chunk;

// 将socket对象赋予用户，赋予用户可操作权限

users[nickname] = socket;

console.log('\033[90m > ' + nickname + ' joined the room\033[39m\n');

broadcast('\033[90m > ' + nickname + ' joined the room\033[39m\n');

}

} else {

// 验证用户为已注册，则输入数据(chunk)为聊天信息 把聊天信息光广播到其他用户里

broadcast('\033[96m > ' + nickname + ':\033[39m ' + chunk+'\n', true);

}

});

**3.3程序测试结果**

（1）首先将服务开启，用webstorm将项目打开，进入到项目目录，输入node server.js命令将服务开启，如图3-2所示。

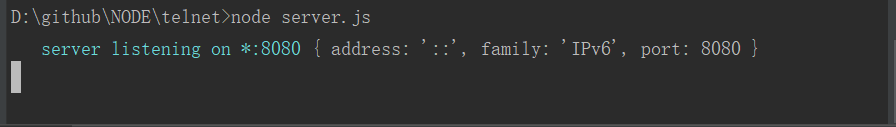


图3-2.服务开启图

（2）Windows键+R打开dos平台（以下说明皆以windows10为例），输入bash命令进入操作界面，并输入telnet + 远程主机IP + 监听端口号连接远程主机，界面如图3-3所示。



图3-3.连接主机图

（3）主机可以通过telnet 127.0.0.1 8080 进入聊天室。进入聊天室后，系统会计算当前聊天室的人数并显示出来，如图3-4所示。

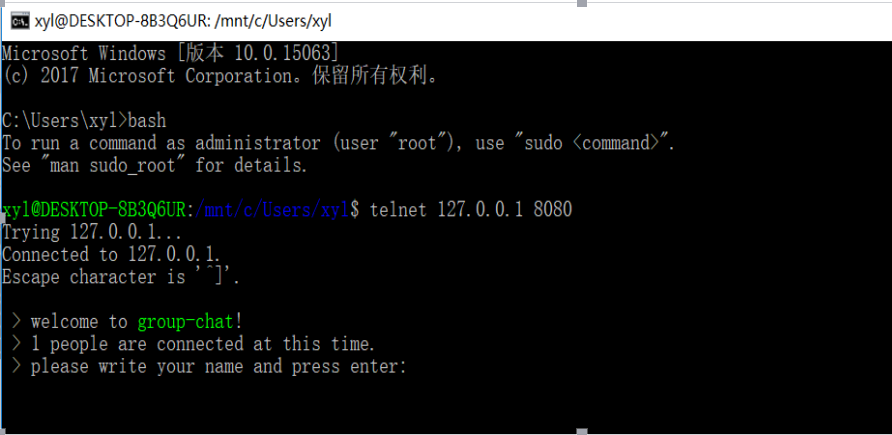


图3-4.进入系统图

（4）进入聊天室后，首先会让用户输入用户名，比如user1，可以多个用户进行连接分别设置用户名为user2，user3，每当一个用户输入完成用户名，如果用户名重复，系统会给出提示，并要求用户重新输入，如图3-5所示。

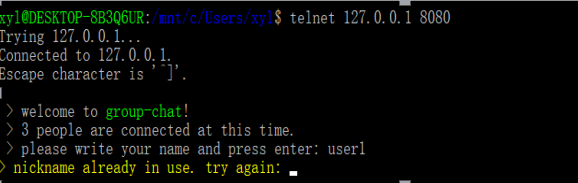


图3-5.用户名重复图

（5）当输入用户名满足要求后，回车，系统会将消息发送给聊天室内的所有成员，提醒有新成员加入了聊天室，如图3-6所示。



图3-6.用户登入图

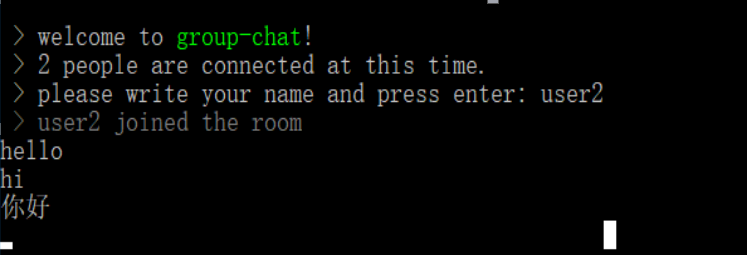
（6）接下来用户就可以进行愉快的聊天了，聊天的信息内容和用户名会发送给聊天室内除了本人之外的其它所有成员，具体情况如图3-7和图3-8所示。

图3-7.发送消息图

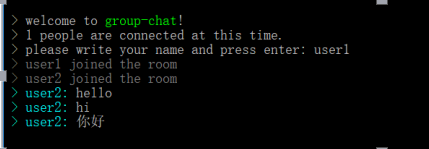


图3-8.接受消息图

（7）当用户关闭dos平台后，便退出了聊天室，这时系统会通知聊天室内成员，该用户已离开当前聊天室，如图3-9所示。

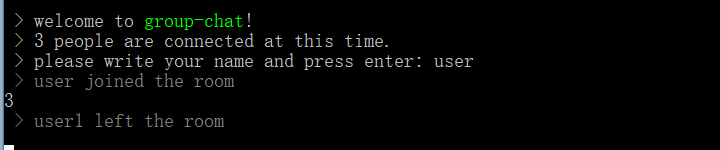


图3-9.退出系统图

**4结束语**

**4.1课程设计的感想**

课程设计可以检验学生对知识的掌握程度。同时更反映了一个学生对课程设计的态度，对待任何事都要认真，可以不会，可以做的不是最好的，但是一定要尽自己最大的努力去完成一件哪怕再小的事。在这次课程设计中学到了很多新的知识，同时也巩固了之前学过知识。对课程设计中所用到的node和telnet有了很好的认识，对网络嗅探有了更深的了解。虽然，最后根据要求完成了任务，但期间还是出现了种种的问题，比如启动服务的的时候程序报错了。同时为了更好的完成任务还查阅了好多资料，不懂的就到处请教，去网上搜索相关的资料来看，锻炼了自己查阅资料的能力。这次的课程设计是我们付出劳动的成果的彰显,也是我们辛苦过程的体现。这种不断上进,认真一致的心态也必将导致一个人在生活和学习的各个方面做的很完美,那种追求的锲而不舍的过程是相同的,这就是一种优良的品质,它将指引着一个人意气风发,更好走好自己的每一步。 在今后的学习中,一定要戒骄戒躁,态度端正,虚心认。要永远的记住一句话:态度决定一切。 总之，在这次机械原理课程设计中，我们受益匪浅。我们会将这次设计中学到的知识道理运用于未来的生活工作中。

**4.2开发过程中的难题与收获**

在此次课程设计的完成过程中，遇到的难题有以下几种：

1. 对telnet协议不熟悉，开发过程中遇到了很多瓶颈。
2. 在利用node的net模块时，对一些api不是很熟悉，由此引发了一些bug。
3. 开发过程中，发现不同系统对程序的解释有所不同，在这方面停留了很长时间。

虽然说开发过程中问题有很多，但解决问题之后得到的收获还是挺令人愉悦的，具体收获有以下几种：

1. 熟悉了telnet协议的工作原理，对于网络这一块有了一些更加深入的学习。
2. 对于node模块编程这一块，得到了一些开发经验。
3. 对于未知的问题，提高了如何去解决它的能力。

**5参考文献**

[1]Pedro Teixeira.node.js高级编程[M].北京：清华大学出版社2015,12,20.

[2]朴灵.深入浅出Node.js[M]..北京：人民邮电出版社,2014,12,29.

[3]李晶.NodeJS——服务器端Javascript运行环境[J].程序员,2010,12,29.

[4]BY Void.Node.Js开发指南[M].北京：人民邮电出版社.2012.7.

**附录：程序源代码**

//引入net模块

const net = require('net');

// 共享状态

//用户计数器

let count = 0;

//用户列表

let users = {};

let server = net.createServer(function (socket) {

count++;

let nickname;

//设置编码格式

socket.setEncoding('utf8');

// 页面tip

socket.write(

'\n > welcome to \033[92mgroup-chat\033[39m!'

+ '\n > ' + count + ' people are connected at this time.'

+ '\n > please write your name and press enter: '

);

// 当用户退出时，进行广播通知

let broadcast = (msg, exceptMyself) => {

for (var i in users) {

if (!exceptMyself || i != nickname) {

users[i].write(msg);

}

}

};

// 监听用户行为作出处理

socket.on('data', function (chunk) {

socket.setEncoding('utf8');

// 删除回车符

chunk = chunk.replace('\r\n', '');

if (!nickname) {

if (users[chunk]) {

socket.write('\033[93m> nickname already in use. try again:\033[39m ');

return;

} else {

nickname = chunk;

// 将socket对象赋予用户，赋予用户可操作权限

users[nickname] = socket;

console.log('\033[90m > ' + nickname + ' joined the room\033[39m\n');

broadcast('\033[90m > ' + nickname + ' joined the room\033[39m\n');

}

} else {

// 验证用户为已注册，则输入数据(chunk)为聊天信息 把聊天信息光广播到其他用户里

broadcast('\033[96m > ' + nickname + ':\033[39m ' + chunk+'\n', true);

}

});

socket.on('error', function (err) {

console.log(err)

});

// 当其中某个用户断开连接时，需要清除数据

socket.on('close', function () {

count--;

broadcast('\033[90m > ' + nickname + ' left the room\033[39m\n');

delete users[nickname];

});

});

server.listen(8080, function () {

console.log('\033[96m server listening on \*:8080\033[39m', server.address());

});

server.on('error',function(e){

if (e.code === 'EADDRINUSE') {

console.log('Address in use, retrying...');

setTimeout(function() {

server.close();

server.listen(8080);

}, 1000);

}

});

Body-parser模块：

var deprecate = require('depd')('body-parser')

var parsers = Object.create(null)

/\*\*

\* @typedef Parsers

\* @type {function}

\* @property {function} json

\* @property {function} raw

\* @property {function} text

\* @property {function} urlencoded

\*/

/\*\*

\* Module exports.

\* @type {Parsers}

\*/

exports = module.exports = deprecate.function(bodyParser,

'bodyParser: use individual json/urlencoded middlewares')

/\*\*

\* JSON parser.

\* @public

\*/

Object.defineProperty(exports, 'json', {

configurable: true,

enumerable: true,

get: createParserGetter('json')

})

/\*\*

\* Raw parser.

\* @public

\*/

Object.defineProperty(exports, 'raw', {

configurable: true,

enumerable: true,

get: createParserGetter('raw')

})

/\*\*

\* Text parser.

\* @public

\*/

Object.defineProperty(exports, 'text', {

configurable: true,

enumerable: true,

get: createParserGetter('text')

})

Object.defineProperty(exports, 'urlencoded', {

configurable: true,

enumerable: true,

get: createParserGetter('urlencoded')

})

/\*\*

\* Create a middleware to parse json and urlencoded bodies.

\*

\* @param {object} [options]

\* @return {function}

\* @deprecated

\* @public

\*/

function bodyParser (options) {

var opts = {}

// exclude type option

if (options) {

for (var prop in options) {

if (prop !== 'type') {

opts[prop] = options[prop]

}

}

}

var \_urlencoded = exports.urlencoded(opts)

var \_json = exports.json(opts)

return function bodyParser (req, res, next) {

\_json(req, res, function (err) {

if (err) return next(err)

\_urlencoded(req, res, next)

})

}

}

/\*\*

\* Create a getter for loading a parser.

\* @private

\*/

function createParserGetter (name) {

return function get () {

return loadParser(name)

}

}

/\*\*

\* Load a parser module.

\* @private

\*/

function loadParser (parserName) {

var parser = parsers[parserName]

if (parser !== undefined) {

return parser

}

// this uses a switch for static require analysis

switch (parserName) {

case 'json':

parser = require('./lib/types/json')

break

case 'raw':

parser = require('./lib/types/raw')

break

case 'text':

parser = require('./lib/types/text')

break

case 'urlencoded':

parser = require('./lib/types/urlencoded')

break

}

// store to prevent invoking require()

return (parsers[parserName] = parser)

}