Python *fcntl.ioctl* Examples

The following are *34* code examples for showing how to use *fcntl.ioctl*. They are extracted from open source Python projects. You can click to vote up the examples you like, or click to vote down the exmaples you don't like. Your votes will be used in our system to extract more high-quality examples.

You may also check out all available functions/classes of the module fcntl , or try the search function .

Example 1

From project *gecko-dev*, under directory *testing/web-platform/tests/webdriver*, in source file *network.py*.

Score: 19

```
def get_interface_ip(ifname):
        sckt = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        return socket.inet_ntoa(fcntl.ioctl(
                sckt.fileno(),
                0x8915,  # SIOCGIFADDR
                struct.pack('256s', ifname[:15])
            )[20:24])
```

Example 2

From project *gecko-dev*, under directory *testing/mozbase/moznetwork/moznetwork*, in source file *moznetwork.py*.

Score: 13

```
def _get_interface_list():
    """Provides a list of available network interfaces
       as a list of tuples (name, ip)"""
    max_iface = 32  # Maximum number of interfaces(Aribtrary)
    bytes = max_iface * 32
    is_32bit = (8 * struct.calcsize("P")) == 32  # Set Architecture
    struct_size = 32 if is_32bit else 40

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        names = array.array('B', '\0' * bytes)
        outbytes = struct.unpack('iL', fcntl.ioctl(
            s.fileno(),
            0x8912,  # SIOCGIFCONF
            struct.pack('iL', bytes, names.buffer_info()[0])
        ))[0]
        namestr = names.tostring()
        return [(namestr[i:i + 32].split('\0', 1)[0],
                socket.inet_ntoa(namestr[i + 20:i + 24]))\
                for i in range(0, outbytes, struct_size)]

    except IOError:
        raise NetworkError('Unable to call ioctl with SIOCGIFCONF')
```

Example 3

From project *cloudman-master*, under directory *cm/clouds*, in source file *dummy.py*.

Score: 13

```
def _getIpAddress(self, ifname):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            ip = socket.inet_ntoa(fcntl.ioctl(
                s.fileno(),
                0x8915,  # SIOCGIFADDR
```

```
                struct.pack('256s', ifname[:15])
            )[20:24])
        except IOError:
            return None
        return ip
```

Example 4

From project *smartdns*, under directory *bin*, in source file *sdns.py*.

Score: 10

```
def get_local_ip():
        import sys,socket,fcntl,array,struct
        is_64bits = sys.maxsize > 2**32
        struct_size = 40 if is_64bits else 32
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        max_possible = 8 # initial value
        while True:
                bytes = max_possible * struct_size
                names = array.array('B', '\0' * bytes)
                outbytes = struct.unpack('iL', fcntl.ioctl(
                        s.fileno(),
                        0x8912,  # SIOCGIFCONF
                        struct.pack('iL', bytes, names.buffer_info()[0])
                ))[0]
                if outbytes == bytes:
                        max_possible *= 2
                else:
                        break
        namestr = names.tostring()
        return [(namestr[i:i+16].split('\0', 1)[0],
                socket.inet_ntoa(namestr[i+20:i+24]))
                for i in range(0, outbytes, struct_size)]
```

Example 5

From project *bitcoin*, under directory *qa/rpc-tests*, in source file *netutil.py*.

Score: 10

```
def all_interfaces():
    '''
    Return all interfaces that are up
    '''
    is_64bits = sys.maxsize > 2**32
    struct_size = 40 if is_64bits else 32
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    max_possible = 8 # initial value
    while True:
        bytes = max_possible * struct_size
        names = array.array('B', '\0' * bytes)
        outbytes = struct.unpack('iL', fcntl.ioctl(
            s.fileno(),
            0x8912,  # SIOCGIFCONF
            struct.pack('iL', bytes, names.buffer_info()[0])
        ))[0]
        if outbytes == bytes:
            max_possible *= 2
        else:
            break
    namestr = names.tostring()
    return [(namestr[i:i+16].split('\0', 1)[0],
             socket.inet_ntoa(namestr[i+20:i+24]))
            for i in range(0, outbytes, struct_size)]
```

Example 6

From project *rlundo-master*, under directory *rlundo*, in source file *pity.py*.

```
def clone_window_size_from(slave_name, from_fd):
    slave_fd = os.open(slave_name, os.O_RDWR)
    try:
        fcntl.ioctl(
            slave_fd,
            termios.TIOCSWINSZ,
            fcntl.ioctl(from_fd, termios.TIOCGWINSZ, " " * 1024)
        )
    finally:
        os.close(slave_fd)
```

Score: 10

Example 7

From project *pyoac*, under directory *pypy/module/fcntl/test*, in source file *test_fcntl.py*.

```
def test_ioctl(self):
        import fcntl
        import array
        import sys, os

        if "linux" in sys.platform:
            TIOCGPGRP = 0x540f
        elif "darwin" in sys.platform or "freebsd6" == sys.platform:
            TIOCGPGRP = 0x40047477
        else:
            skip("don't know how to test ioctl() on this platform")

        raises(TypeError, fcntl.ioctl, "foo")
        raises(TypeError, fcntl.ioctl, 0, "foo")
        #raises(TypeError, fcntl.ioctl, 0, TIOCGPGRP, float(0))
        raises(TypeError, fcntl.ioctl, 0, TIOCGPGRP, 1, "foo")

        if not os.isatty(0):
            skip("stdin is not a tty")

        buf = array.array('h', [0])
        res = fcntl.ioctl(0, TIOCGPGRP, buf, True)
        assert res == 0
        assert buf[0] != 0
        expected = buf.tostring()

        if '__pypy__' in sys.builtin_module_names or sys.version_info >= (2,5):
            buf = array.array('h', [0])
            res = fcntl.ioctl(0, TIOCGPGRP, buf)
            assert res == 0
            assert buf.tostring() == expected

        res = fcntl.ioctl(0, TIOCGPGRP, buf, False)
        assert res == expected

        raises(TypeError, fcntl.ioctl, 0, TIOCGPGRP, "\x00\x00", True)

        res = fcntl.ioctl(0, TIOCGPGRP, "\x00\x00")
        assert res == expected
```

Score: 10

Example 8

From project *pyoac*, under directory *pypy/lib/py/misc/testing*, in source file *test_terminal.py*.

```
def test_terminal_width():
    """ Dummy test for get_terminal_width
    """
    assert get_terminal_width()
    try:
        import fcntl
    except ImportError:
        py.test.skip('fcntl not supported on this platform')
    def f(*args):
        raise ValueError
```

Score: 10

```
    ioctl = fcntl.ioctl
    fcntl.ioctl = f
    try:
        cols = os.environ.get('COLUMNS', None)
        os.environ['COLUMNS'] = '42'
        assert get_terminal_width() == 41
    finally:
        fcntl.ioctl = ioctl
        if cols:
            os.environ['COLUMNS'] = cols
```

Example 9

From project *hue*, under directory *desktop/core/ext-py/Twisted/twisted/test*, in source file *test_process.py*.

```
def ioctl(self, fd, flags, arg):
        """
        Override C{fcntl.ioctl}. Do nothing.
        """
```

Score: 10

Example 10

From project *Pale-Moon*, under directory *testing/mozbase/mozdevice/mozdevice*, in source file *devicemanager.py*.

```
def getInterfaceIp(self, ifname):
        if os.name != "nt":
            import fcntl
            import struct
            s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            return socket.inet_ntoa(fcntl.ioctl(
                                    s.fileno(),
                                    0x8915,  # SIOCGIFADDR
                                    struct.pack('256s', ifname[:15])
                                    )[20:24])
        else:
            return None
```

Score: 10

Example 11

From project *Pale-Moon*, under directory *testing/mozbase/moznetwork/moznetwork*, in source file *moznetwork.py*.

```
def _get_interface_list():
    """Provides a list of available network interfaces
       as a list of tuples (name, ip)"""
    max_iface = 32  # Maximum number of interfaces(Aribtrary)
    bytes = max_iface * 32
    is_32bit = (8 * struct.calcsize("P")) == 32  # Set Architecture
    struct_size = 32 if is_32bit else 40

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        names = array.array('B', '\0' * bytes)
        outbytes = struct.unpack('iL', fcntl.ioctl(
            s.fileno(),
            0x8912,  # SIOCGIFCONF
            struct.pack('iL', bytes, names.buffer_info()[0])
        ))[0]
        namestr = names.tostring()
        return [(namestr[i:i + 32].split('\0', 1)[0],
                socket.inet_ntoa(namestr[i + 20:i + 24]))\
                for i in range(0, outbytes, struct_size)]

    except IOError:
        raise NetworkError('Unable to call ioctl with SIOCGIFCONF')
```

Score: 10

Example 12

From project *Pale-Moon*, under directory *testing/mozbase/mozhttpd/mozhttpd*, in source file *iface.py*.

Score: 10

```
def _get_interface_ip(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            0x8915,  # SIOCGIFADDR
            struct.pack('256s', ifname[:15])
            )[20:24])
```

Example 13

From project *squeal*, under directory *src/squeal*, in source file *util.py*.

Score: 10

```
def all_interfaces():
    max_possible = 128  # arbitrary. raise if needed.
    bytes = max_possible * 32
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    names = array.array('B', '\0' * bytes)
    outbytes = struct.unpack('iL', fcntl.ioctl(
        s.fileno(),
        0x8912,  # SIOCGIFCONF
        struct.pack('iL', bytes, names.buffer_info()[0])
    ))[0]
    namestr = names.tostring()
    return [namestr[i:i+32].split('\0', 1)[0] for i in range(0, outbytes, 32)]
```

Example 14

From project *Byzantium*, under directory *captive_portal*, in source file *fake_dns.py*.

Score: 10

```
def get_ip_address(ifname):
  # > LOOK
  # You are in a maze of twisty passages, all alike.
  # > GO WEST
  s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  try:
    # It is dark here.  You are likely to be eaten by a grue.
    # > _
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])
  except:
    return None

# Display usage information to the user.
```

Example 15

From project *tools-master*, under directory *nsdtool/project*, in source file *network.py*.

Score: 10

```
def get_ip_address(self, ifname):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        try:
            ip = socket.inet_ntoa(fcntl.ioctl(
                s.fileno(),
```

```
                0x8915,
                struct.pack('256s', ifname[:15].encode('utf-8')))[20:24])
        except OSError:
            print("Wrong interface name: " + ifname + "\ncheck config.ini")
            sys.exit(0)

        return ip
```

Example 16

From project *VTK*, under directory *ThirdParty/Twisted/twisted/pair*, in source file *tuntap.py*.

```
def ioctl(fd, opt, arg=None, mutate_flag=None):
        """
        @see: L{fcntl.ioctl}
        """
```

Score: 10

Example 17

From project *git-guilt-master*, under directory *git_guilt*, in source file *guilt.py*.

```
def _get_tty_width(self):
        if not self._is_tty:
            return Formatter._default_width

        try:
            (_, w, _, _) = struct.unpack(
                'HHHH',
                fcntl.ioctl(
                    sys.stdout.fileno(),
                    termios.TIOCGWINSZ,
                    struct.pack('HHHH', 0, 0, 0, 0)
                )
            )
        except IOError:
            return Formatter._default_width

        if 0 < w:
            return w
        else:
            return Formatter._default_width
```

Score: 10

Example 18

From project *mmc*, under directory *pulse2/services/pulse2*, in source file *utils.py*.

```
def get_ip_address(ifname):
    """ TODO: IPv6
    """
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname)
    )[20:24])
```

Score: 10

Example 19

From project *mmc*, under directory *core/agent/mmc/support*, in source file *mmctools.py*.

```
def localifs():
    """
    Used to get a list of the up interfaces and associated IP addresses
    on this machine (linux only).

    Returns:
        List of interface tuples.  Each tuple consists of
        (interface name, interface IP)
    """

    SIOCGIFCONF = 0x8912
    MAXBYTES = 8096

    arch = platform.architecture()[0]

    if arch == '32bit':
        var1 = 32
        var2 = 32
    elif arch == '64bit':
        var1 = 16
        var2 = 40
    else:
        raise OSError("Unknown architecture: %s" % arch)

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    names = array.array('B', '\0' * MAXBYTES)
    outbytes = struct.unpack('iL', fcntl.ioctl(
        sock.fileno(),
        SIOCGIFCONF,
        struct.pack('iL', MAXBYTES, names.buffer_info()[0])
        ))[0]

    namestr = names.tostring()
    return [(namestr[i:i+var1].split('\0', 1)[0], socket.inet_ntoa(namestr[i+20:i+24])) \
            for i in xrange(0, outbytes, var2)]
```

Score: 10

Example 20

From project *pygstlib*, under directory *util*, in source file *localip.py*.

```
def get_ip_for_if(ifname):
    """Return IP for given interface name
    """
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        return socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            0x8915,  # SIOCGIFADDR
            struct.pack('256s', ifname[:15])
            )[20:24])
    except IOError:
        return None
#
```

Score: 10

Example 21

From project *LANBox*, under directory *cli/servermanager*, in source file *util.py*.

```
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])
```

Score: 10

Example 22

From project *LANBox*, under directory *codadmin/cod*, in source file *util.py*.

Score: 10

```
def get_ip_address_by_iface(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])
```

Example 23

From project *python-for-android-master*, under directory *python-modules/pybluez/examples/advanced*, in source file *l2-unreliable-client.py*.

Score: 10

```
def __get_acl_conn_handle(sock, addr):
    hci_fd = sock.fileno()
    reqstr = struct.pack( "6sB17s", bt.str2ba(addr), bt.ACL_LINK, "\0" * 17)
    request = array.array( "c", reqstr )
    fcntl.ioctl( hci_fd, bt.HCIGETCONNINFO, request, 1 )
    handle = struct.unpack("8xH14x", request.tostring())[0]
    return handle
```

Example 24

From project *python-for-android-master*, under directory *python-modules/pybluez/bluetooth*, in source file *bluez.py*.

Score: 10

```
def _get_acl_conn_handle (hci_sock, addr):
    hci_fd = hci_sock.fileno ()
    reqstr = struct.pack ("6sB17s", _bt.str2ba (addr),
            _bt.ACL_LINK, "\0" * 17)
    request = array.array ("c", reqstr)
    try:
        fcntl.ioctl (hci_fd, _bt.HCIGETCONNINFO, request, 1)
    except IOError, e:
        raise BluetoothError ("There is no ACL connection to %s" % addr)

    # XXX should this be "<8xH14x"?
    handle = struct.unpack ("8xH14x", request.tostring ())[0]
    return handle
```

Example 25

From project *alignak-master*, under directory *libexec/discovery*, in source file *SAN_discover_runner.py*.

Score: 10

```
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])
```

Example 26

From project *sonospy*, under directory *sonospy/gui*, in source file *nowPlayingTab.py*.

```
def get_interface_ip(ifname):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        return socket.inet_ntoa(fcntl.ioctl(
                        s.fileno(),
                        0x8915,  # SIOCGIFADDR
                        struct.pack('256s', ifname[:15])
                )[20:24])
```

Score: 10

Example 27

From project *openmoko-gsoc2008*, under directory *fsod/src/subsystems/Python/onetworkd*, in source file *network.py*.

```
def ipAddress4( self ):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        return socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            0x8915,  # SIOCGIFADDR
            struct.pack('256s', self._name[:15])
        )[20:24])

#============================================================================#
```

Score: 10

Example 28

From project *infernal-twin-master*, under directory *old_CLI_project*, in source file *get_iface.py*.

```
def all_interfaces():
    is_64bits = sys.maxsize > 2**32
    struct_size = 40 if is_64bits else 32
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    max_possible = 8 # initial value
    while True:
        bytes = max_possible * struct_size
        names = array.array('B', '\0' * bytes)
        outbytes = struct.unpack('iL', fcntl.ioctl(
            s.fileno(),
            0x8912,  # SIOCGIFCONF
            struct.pack('iL', bytes, names.buffer_info()[0])
        ))[0]
        if outbytes == bytes:
            max_possible *= 2
        else:
            break
    namestr = names.tostring()
    return [(namestr[i:i+16].split('\0', 1)[0],
             socket.inet_ntoa(namestr[i+20:i+24]))
            for i in range(0, outbytes, struct_size)]
```

Score: 10

Example 29

From project *nova*, under directory *tools/esx*, in source file *guest_tool.py*.

```
def _get_linux_network_adapters():
    """Get the list of Linux network adapters."""
    import fcntl
    max_bytes = 8096
    arch = platform.architecture()[0]
    if arch == ARCH_32_BIT:
        offset1 = 32
        offset2 = 32
    elif arch == ARCH_64_BIT:
        offset1 = 16
        offset2 = 40
    else:
        raise OSError(_("Unknown architecture: %s") % arch)
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Score: 8

```
    names = array.array('B', '\0' * max_bytes)
    outbytes = struct.unpack('iL', fcntl.ioctl(
        sock.fileno(),
        0x8912,
        struct.pack('iL', max_bytes, names.buffer_info()[0])))[0]
    adapter_names = [names.tostring()[n_cnt:n_cnt + offset1].split('\0', 1)[0]
                     for n_cnt in xrange(0, outbytes, offset2)]
    network_adapters = []
    for adapter_name in adapter_names:
        ip_address = socket.inet_ntoa(fcntl.ioctl(
            sock.fileno(),
            0x8915,
            struct.pack('256s', adapter_name))[20:24])
        subnet_mask = socket.inet_ntoa(fcntl.ioctl(
            sock.fileno(),
            0x891b,
            struct.pack('256s', adapter_name))[20:24])
        raw_mac_address = '%012x' % _bytes2int(fcntl.ioctl(
            sock.fileno(),
            0x8927,
            struct.pack('256s', adapter_name))[18:24])
        mac_address = ":".join([raw_mac_address[m_counter:m_counter + 2]
            for m_counter in range(0, len(raw_mac_address), 2)]).lower()
        network_adapters.append({'name': adapter_name,
                                 'mac-address': mac_address,
                                 'ip-address': ip_address,
                                 'subnet-mask': subnet_mask})
    return network_adapters
```

Example 30

From project *XacCRM*, under directory *tools*, in source file *misc.py*.

Score: 8

```
def detect_ip_addr():
    """Try a very crude method to figure out a valid external
       IP or hostname for the current machine. Don't rely on this
       for binding to an interface, but it could be used as basis
       for constructing a remote URL to the server.
    """
    def _detect_ip_addr():
        from array import array
        import socket
        from struct import pack, unpack

        try:
            import fcntl
        except ImportError:
            fcntl = None

        ip_addr = None

        if not fcntl: # not UNIX:
            host = socket.gethostname()
            ip_addr = socket.gethostbyname(host)
        else: # UNIX:
            # get all interfaces:
            nbytes = 128 * 32
            s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            names = array('B', '\0' * nbytes)
            #print 'names: ', names
            outbytes = unpack('iL', fcntl.ioctl( s.fileno(), 0x8912, pack('iL', nbytes, names.buffer_inf
            namestr = names.tostring()

            # try 64 bit kernel:
            for i in range(0, outbytes, 40):
                name = namestr[i:i+16].split('\0', 1)[0]
                if name != 'lo':
                    ip_addr = socket.inet_ntoa(namestr[i+20:i+24])
                    break

            # try 32 bit kernel:
            if ip_addr is None:
                ifaces = filter(None, [namestr[i:i+32].split('\0', 1)[0] for i in range(0, outbytes, 32)
```

```
                for ifname in [iface for iface in ifaces if iface != 'lo']:
                    ip_addr = socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, pack('256s', ifname[:15]))
                    break

        return ip_addr or 'localhost'

    try:
        ip_addr = _detect_ip_addr()
    except:
        ip_addr = 'localhost'
    return ip_addr

# RATIONALE BEHIND TIMESTAMP CALCULATIONS AND TIMEZONE MANAGEMENT:
#  The server side never does any timestamp calculation, always
#  sends them in a naive (timezone agnostic) format supposed to be
#  expressed within the server timezone, and expects the clients to
#  provide timestamps in the server timezone as well.
#  It stores all timestamps in the database in naive format as well,
#  which also expresses the time in the server timezone.
#  For this reason the server makes its timezone name available via the
#  common/timezone_get() rpc method, which clients need to read
#  to know the appropriate time offset to use when reading/writing
#  times.
```

Example 31

From project *openerp-ktv*, under directory *openerp/tools*, in source file *misc.py*.

Score: 8

```
def detect_ip_addr():
    """Try a very crude method to figure out a valid external
       IP or hostname for the current machine. Don't rely on this
       for binding to an interface, but it could be used as basis
       for constructing a remote URL to the server.
    """
    def _detect_ip_addr():
        from array import array
        from struct import pack, unpack

        try:
            import fcntl
        except ImportError:
            fcntl = None

        ip_addr = None

        if not fcntl: # not UNIX:
            host = socket.gethostname()
            ip_addr = socket.gethostbyname(host)
        else: # UNIX:
            # get all interfaces:
            nbytes = 128 * 32
            s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            names = array('B', '\0' * nbytes)
            #print 'names: ', names
            outbytes = unpack('iL', fcntl.ioctl( s.fileno(), 0x8912, pack('iL', nbytes, names.buffer_inf
            namestr = names.tostring()

            # try 64 bit kernel:
            for i in range(0, outbytes, 40):
                name = namestr[i:i+16].split('\0', 1)[0]
                if name != 'lo':
                    ip_addr = socket.inet_ntoa(namestr[i+20:i+24])
                    break

            # try 32 bit kernel:
            if ip_addr is None:
                ifaces = filter(None, [namestr[i:i+32].split('\0', 1)[0] for i in range(0, outbytes, 32)]

                for ifname in [iface for iface in ifaces if iface != 'lo']:
                    ip_addr = socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, pack('256s', ifname[:15]))
                    break

        return ip_addr or 'localhost'
```

```
    try:
        ip_addr = _detect_ip_addr()
    except Exception:
        ip_addr = 'localhost'
    return ip_addr

# RATIONALE BEHIND TIMESTAMP CALCULATIONS AND TIMEZONE MANAGEMENT:
#  The server side never does any timestamp calculation, always
#  sends them in a naive (timezone agnostic) format supposed to be
#  expressed within the server timezone, and expects the clients to
#  provide timestamps in the server timezone as well.
#  It stores all timestamps in the database in naive format as well,
#  which also expresses the time in the server timezone.
#  For this reason the server makes its timezone name available via the
#  common/timezone_get() rpc method, which clients need to read
#  to know the appropriate time offset to use when reading/writing
#  times.
```

Example 32

From project *kunai-master*, under directory *kunai*, in source file *util.py*.

Score: 7

```
def get_ip_address(ifname):
   s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
   return socket.inet_ntoa(fcntl.ioctl(
      s.fileno(),
      0x8915,  # SIOCGIFADDR
      struct.pack('256s', ifname[:15])
   )[20:24])
```

Example 33

From project *mmc*, under directory *pulse2/services/pulse2/launcher*, in source file *tcp_sproxy.py*.

Score: 5

```
def establishProxy(client, requestor_ip, requested_port):
    """
    Establish a TCP connection to client using our proxy
    """
    client = pulse2.launcher.utils.setDefaultClientOptions(client)
    """
    client['client_check'] = getClientCheck(client)
    client['server_check'] = getServerCheck(client)
    client['action'] = getAnnounceCheck('vnc')
    """
    def generate_auth_key():
        import random
        chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$-_.+!*()'
        size = 15
        return ''.join(random.choice(chars) for x in range(size))

    if LauncherConfig().create_web_proxy:
        auth_key = generate_auth_key()
    else:
        auth_key = '-'

    proxy_port, local_port = allocate_port_couple()
    # Built "exec" command
    real_command = [
        LauncherConfig().tcp_sproxy_path,
        requestor_ip,
        client['host'],
        requested_port,
        ','.join(client['transp_args']),
        str(proxy_port),
        str(local_port),
        str(LauncherConfig().tcp_sproxy_establish_delay),
        str(LauncherConfig().tcp_sproxy_connect_delay),
        str(LauncherConfig().tcp_sproxy_session_lenght),
        client['shortname'],
        auth_key,
```

```
    ]

    # Built "thru" command
    thru_command_list  = [LauncherConfig().ssh_path]
    for option in client['transp_args']:
        thru_command_list += ['-o', option]
    thru_command_list += [client['host']]

    command_list = [
        LauncherConfig().wrapper_path,
        '--max-log-size',
        str(LauncherConfig().wrapper_max_log_size),
        #'--max-exec-time', # FIXME: wrapper_timeout missing in function signature :/
        #str(wrapper_timeout),
        '--exec',
        SEPARATOR.join(real_command),
        '--thru',
        SEPARATOR.join(thru_command_list),
        '--no-wrap',
        '--only-stdout',
        '--remove-empty-lines',
        '--exec-server-side'
    ]

    # from {'a': 'b', 'c: 'd'} to 'a=b,c=d'
    if client['client_check']:
        command_list += ['--check-client-side', ','.join(map((lambda x: '='.join(x)), client['client_che
    if client['server_check']:
        command_list += ['--check-server-side', ','.join(map((lambda x: '='.join(x)), client['server_che
    if client['action']:
        command_list += ['--action', client['action']]

    proxy = proxyProtocol()
    twisted.internet.reactor.spawnProcess(
        proxy,
        command_list[0],
        map(lambda(x): x.encode('utf-8', 'ignore'), command_list),
        None, # env
        None, # path
        None, # uid
        None, # gid
        None, # usePTY
        { 0: "w", 1: 'r', 2: 'r' } # FDs: not closing STDIN (might be used)
    )

    def get_ip_address(ifname):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        return socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            0x8915,  # SIOCGIFADDR
            struct.pack('256s', ifname[:15])
        )[20:24])

    def get_all_interfaces():
        """
        Return list of all available interfaces with IP
        @see: https://gist.github.com/pklaus/289646
        """
        def format_ip(addr):
            return '.'.join([str(ord(x)) for x in addr])
        max_possible = 128  # arbitrary. raise if needed.
        bytes = max_possible * 32
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        names = array.array('B', '\0' * bytes)
        outbytes = struct.unpack('iL', fcntl.ioctl(
            s.fileno(),
            0x8912,  # SIOCGIFCONF
            struct.pack('iL', bytes, names.buffer_info()[0])
        ))[0]
        namestr = names.tostring()
        lst = []
        for i in range(0, outbytes, 40):
            name = namestr[i:i+16].split('\0', 1)[0]
            ip   = format_ip(namestr[i+20:i+24])
            if not ip.startswith('127.'):
                lst.append({
                    'name': name,
                    'ip': ip,
```

```
                })
        return lst

    def parse_result():
        if LauncherConfig().tcp_sproxy_host:
            tcp_sproxy_host = LauncherConfig().tcp_sproxy_host
        else:
            # Take the first network interface
            logging.getLogger().info('tcp_sproxy_host param was not specified in launcher config')
            available_interfaces = get_all_interfaces()
            if available_interfaces:
                tcp_sproxy_host = available_interfaces[0]['ip']
                logging.getLogger().info('Using %(name)s %(ip)s as IP address for tcp_sproxy_host' % ava
            else:
                raise Exception('No IP found for tcp_sproxy_host')
        return LauncherConfig().name, tcp_sproxy_host, proxy_port, auth_key
    # Waiting to establish the proxy
    ret = task.deferLater(twisted.internet.reactor, 2, parse_result)
    logging.getLogger().debug('about to execute ' + ' '.join(command_list))
    return ret
```

Example 34

From project *coherence*, under directory *coherence/upnp/core*, in source file *utils.py*.

Score: 5

```
def get_ip_address(ifname):
    """
    determine the IP address by interface name

    http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/439094
    (c) Paul Cannon
    Uses the Linux SIOCGIFADDR ioctl to find the IP address associated
    with a network interface, given the name of that interface, e.g. "eth0".
    The address is returned as a string containing a dotted quad.

    Updated to work on BSD. OpenBSD and OSX share the same value for
    SIOCGIFADDR, and its likely that other BSDs do too.

    Updated to work on Windows,
    using the optional Python module netifaces
    http://alastairs-place.net/netifaces/

    Thx Lawrence for that patch!
    """

    if have_netifaces:
        if ifname in netifaces.interfaces():
            iface = netifaces.ifaddresses(ifname)
            ifaceadr = iface[netifaces.AF_INET]
            # we now have a list of address dictionaries, there may be multiple addresses bound
            return ifaceadr[0]['addr']
    import sys
    if sys.platform in ('win32','sunos5'):
        return '127.0.0.1'

    from os import uname
    import socket
    import fcntl
    import struct

    system_type = uname()[0]

    if system_type == "Linux":
        SIOCGIFADDR = 0x8915
    else:
        SIOCGIFADDR = 0xc0206921

    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        return socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            SIOCGIFADDR,
            struct.pack('256s', ifname[:15])
        )[20:24])
```

```
    except:
        return '127.0.0.1'
```