

**Buyruq qobiqlari. Buyruq qobig'i tushunchasi.
Qobiqlarning turlari. Bash dasturlash**

Kirish

Oldingi ma'ruzalarda biz operatsion tizimlarning ichki tuzilishi, jarayonlar, oqimlar, xotira, kiritish-chiqarish va tizim chaqiruvlarini ko'rib chiqdik.

Endi foydalanuvchining operatsion tizim bilan o'zaro aloqasining eng muhim vositalaridan biri — **buyruq qobig'i (Shell)** ni ko'rib chiqamiz.

Buyruq qobig'i Linux va Unix-ga o'xshash tizimlar bilan ishlashda ayniqsa muhimdir. U keng qo'llaniladi:

- tizim administratorlari tomonidan;
- dasturchilar tomonidan;
- DevOps-muhandislar tomonidan;
- server ilovalarini ishlab chiquvchilar tomonidan;
- axborot xavfsizligi mutaxassislari tomonidan;
- ma'lumotlarni tahlil qilish mutaxassislari tomonidan;
- takrorlanuvchi amallarni avtomatlashtirishi kerak bo'lgan foydalanuvchilar tomonidan.

Buyruq qobig'i nafaqat alohida buyruqlarni bajarishga, balki murakkab amallar ketma-ketligini avtomatlashtiradigan dasturlar — **shell-skriptlar** yaratishga ham imkon beradi.

Buyruq qobig'i nima?

Buyruq qobig'i (Shell) — bu foydalanuvchiga buyruqlar orqali operatsion tizim bilan o'zaro aloqa qilish uchun interfeys taqdim etuvchi dasturiy qobiqdir.



Buyruq qobig'ining asosiy funksiyalari

Buyruq qobig'i bir nechta muhim funksiyalarni bajaradi.

1. Buyruqlarni bajarish

Masalan:

```
pwd
```

Joriy katalogni ko'rsatadi.

```
ls
```

Katalog tarkibini ko'rsatadi.

```
mkdir projects
```

projects katalogini yaratadi.

2. Dasturlarni ishga tushirish

Shell tashqi dasturlarni ishga tushirishi mumkin.

Masalan:

```
python /home/java/app.py
```

Bu holda Shell `python` dasturini topadi, unga `app.py` argumentini uzatadi va jarayonni ishga tushiradi.

3. Fayllar bilan ishlash

Shell orqali quyidagilarni amalga oshirish mumkin:

- fayllarni yaratish;
- fayllarni o'chirish;
- fayllarni nusxalash;
- fayllarni ko'chirish;
- kirish huquqlarini o'zgartirish;
- fayllarni qidirish.

Masalan:

```
cp file.txt backup.txt
```

Faylni nusxalaydi.

```
mv old.txt new.txt
```

Fayl nomini o'zgartiradi.

```
rm file.txt
```

Faylni o'chiradi.

5. Jarayonlar bilan ishlash

Shell jarayonlarni ishga tushirish va nazorat qilish imkonini beradi.

Masalan:

```
python server.py
```

Jarayonni ishga tushiradi.

Dasturni fon rejimida ishga tushirish uchun:

```
python server.py &
```

Ampersand & qobiqqa jarayon fonda bajarilishi kerakligini bildiradi.

Jarayonlarni ko'rish mumkin:

```
ps
```

yoki:

```
ps aux
```

6. Kiritish va chiqarishni yo'naltirish

Shellning eng kuchli imkoniyatlaridan biri standart oqimlarni yo'naltirishdir.

Unixda odatda quyidagilar mavjud:

```
0 → stdin  
1 → stdout  
2 → stderr
```

stdout ni yo'naltirish

```
ls > files.txt
```

ls buyrug'ining natijasi faylga yoziladi.

Faylga qo'shish

```
ls >> files.txt
```

Yangi ma'lumotlar fayl oxiriga qo'shiladi.

Xatolarni yo'naltirish

```
python app.py 2> errors.txt
```

Xatolar errors.txt ga yoziladi.

stdout va stderr bir vaqtda

```
python app.py > output.txt 2> errors.txt
```

7. Konveyerlar

Konveyer (pipe) bir buyruq natijasini to'g'ridan-to'g'ri boshqasiga uzatish imkonini beradi.

Buning uchun quyidagi belgi ishlatiladi:

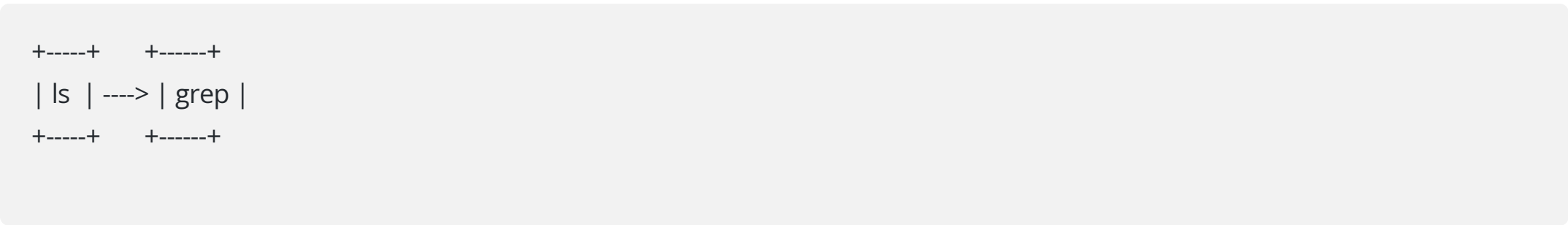
```
|
```

Masalan:

```
ls | grep ".txt"
```

Avval `ls` bajariladi, so'ngra natija `grep` dasturiga uzatiladi.

Sxema:



Zanjirlar yaratish mumkin:

```
cat users.txt | grep "admin" | sort
```

Bu yerda:

- 1. `cat` faylni o'qiydi;
- 2. `grep` `admin` ni o'z ichiga olgan satrlarni qoldiradi;
- 3. `sort` natijani saralaydi.

SXEMA: PIPE BUXOBORDI (SHELL PIPE SCHEMATIC)



ls - RO'YXATGA OLISH
(LISTS DIRECTORY CONTENTS)

grep - QIDIRUV
(SEARCHES FOR PATTERNS)

shell buyrug'i: **ls | grep 'fayl'**

```
$ ls
fayl1.txt
rasm.png
hujjat.pdf
fayl2.csv
$ ls | grep 'fayl'
fayl1.txt
fayl2.csv
```

8. Shell interpretator sifatida

Shell buyruqlar interpretatordir.

Masalan, foydalanuvchi quyidagini kiritadi:

```
echo "Hello"
```

Shell tahlil qiladi:

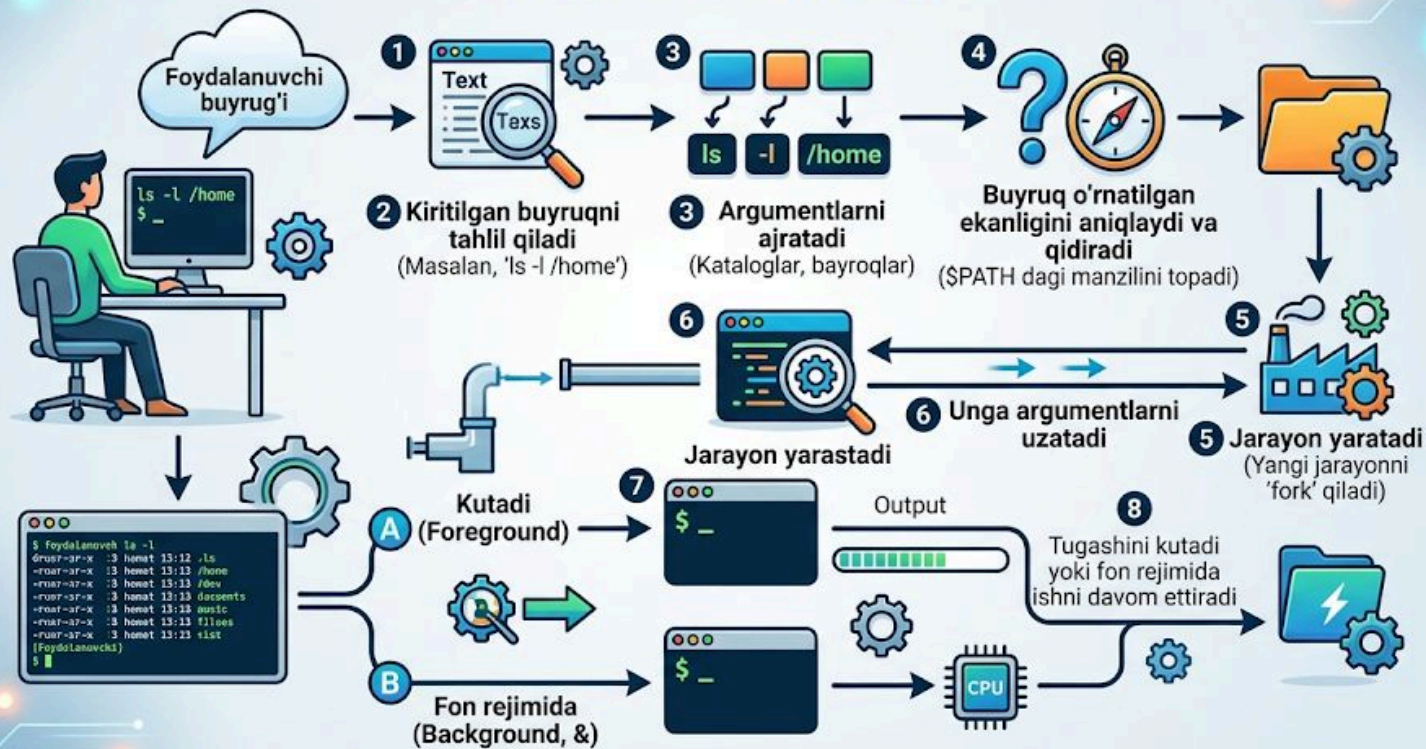
- `echo` buyrug'ini;
- `"Hello"` argumentini.

So'ngra tegishli amal bajariladi.

Tashqi dasturlar bilan ishlashda Shell odatda:

1. kiritilgan buyruqni tahlil qiladi;
2. argumentlarni ajratadi;
3. buyruq o'rnatilgan ekanligini aniqlaydi;
4. kerak bo'lganda bajariladigan faylni qidiradi;
5. jarayon yaratadi;
6. unga argumentlarni uzatadi;
7. tugashini kutadi yoki fon rejimida ishlashda ishni davom ettiradi.

SHELLNING TASHQI DASTURLAR BILAN ISHLASH JARAYONI



Interaktiv qobiq

Foydalanuvchi terminalni ochganda, odatda interaktiv Shell ishga tushiriladi.

Masalan:

```
user@computer:~$
```

Bu yerda:

- `user` — foydalanuvchi nomi;
- `computer` — kompyuter nomi;
- `~` — uy katalogi;
- `$` — oddiy foydalanuvchi uchun taklif belgisi.

Unix-ga o'xshash tizimlarda root uchun ko'pincha quyidagi ishlatiladi:

```
root@computer:~#
```

Buyruq qobiqlarining turlari

Ko'plab qobiqlar mavjud.

Eng mashhur qobiqlar:

- sh;
- Bash;
- Zsh;
- Ksh;
- Fish;
- Csh;
- Tcsh;
- PowerShell;
- Command Prompt (cmd.exe).

BUYRUQ QOBIQ'LARINING TURLARI

Ko'plab qobiqlar mavjud. Eng mashhur qobiqlar:



Bash-skript nima?

Bash-skript — Bash buyruqlari ketma-ketligini o'z ichiga olgan matnli fayl.

Masalan:

```
#!/bin/bash  
  
echo "Hello, World!"
```

Odatda fayl quyidagi kengaytmaga ega bo'ladi:

```
.sh
```

Masalan:

```
backup.sh
```

`.sh` kengaytmasi majburiy emas. Faylning tarkibi va bajarish huquqlari muhimroq.

Shebang

Skriptning birinchi satri ko'pincha quyidagicha ko'rinadi:

```
#!/bin/bash
```

Bu **shebang** deb ataladi.

U tizimga faylni bajarish uchun qaysi interpretatordan foydalanish kerakligini bildiradi.

Masalan:

```
#!/bin/bash
```

Bash dan foydalanadi.

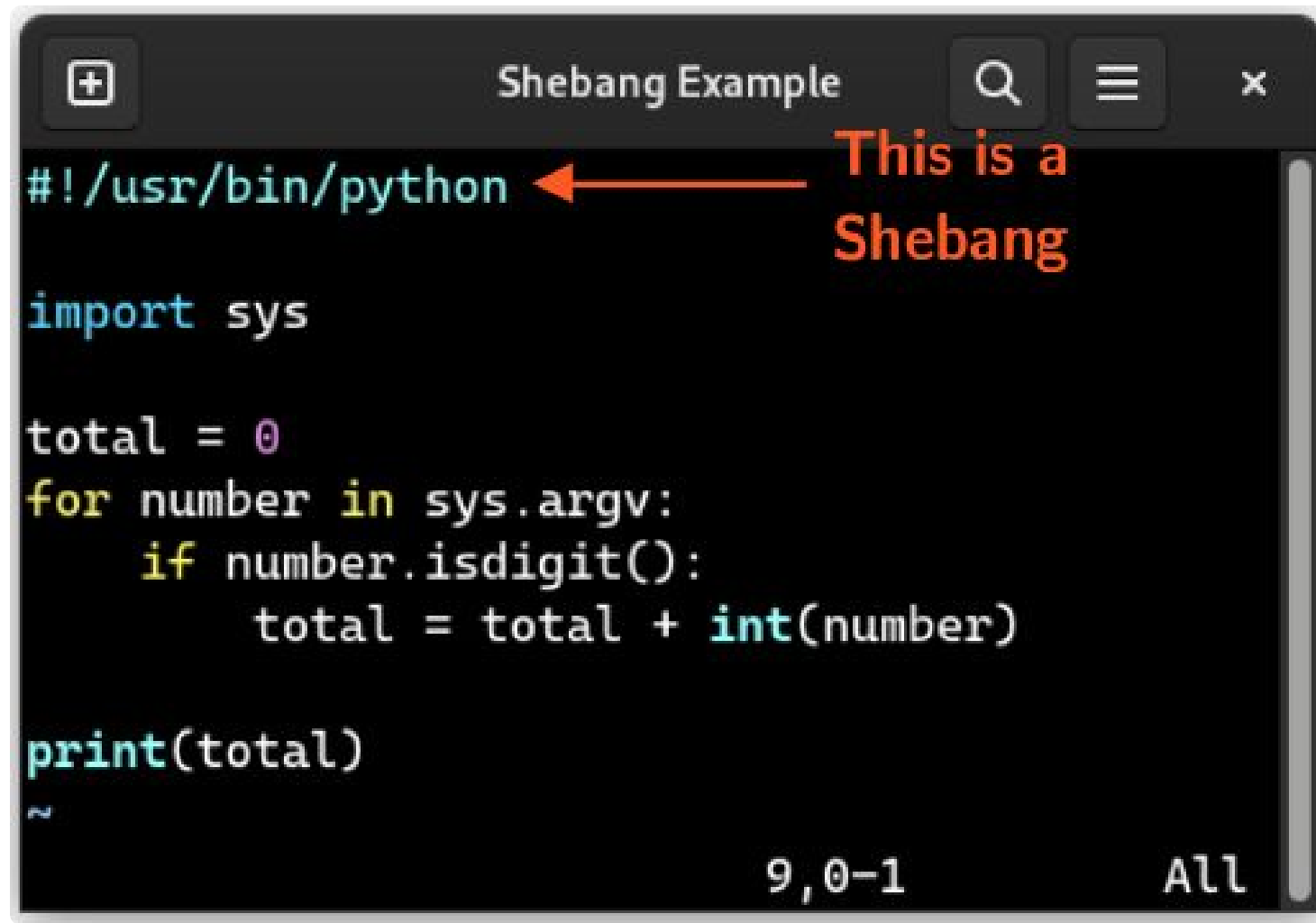
Quyidagini ishlatish mumkin:

```
#!/bin/sh
```

yoki:

```
#!/usr/bin/env bash
```

Oxirgi variant `PATH` muhit o'zgaruvchisi orqali Bash ni qidiradi.



```
#!/usr/bin/python  
  
import sys  
  
total = 0  
for number in sys.argv:  
    if number.isdigit():  
        total = total + int(number)  
  
print(total)  
~
```

9, 0-1 All

Bash-skript

Fayl yaratamiz:

```
hello.sh
```

Tarkibi:

```
#!/bin/bash
```

```
echo "Hello, World!"
```

Faylni bajariladigan qilamiz:

```
chmod +x hello.sh
```

Ishga tushiramiz:

```
./hello.sh
```

Natija:

```
Hello, World!
```

echo buyrug'i

echo buyrug'i matn chiqarish uchun ishlatiladi.

```
echo "Hello"
```

Bir nechta satr chiqarish mumkin:

```
echo "Name: Ahmad"
```

```
echo "Age: 30"
```

O'zgaruvchilar qiymatini chiqarish mumkin:

```
name="Ahmad"
```

```
echo "Hello, $name"
```

Natija:

```
Hello, Ahmad
```

Bash o'zgaruvchilari

Bashda o'zgaruvchilar turini ko'rsatmasdan yaratiladi:

```
name="Ahmad"  
age=30
```

Muhim:

```
name="Ahmad"
```

emas:

```
name = "Ahmad"
```

Oddiy tayinlashda `=` atrofida bo'shliqlar ruxsat etilmaydi.

Qiymatni olish:

```
echo "$name"
```

Misol:

```
#!/bin/bash  
  
name="Ahmad"  
age=30  
  
echo "Name: $name"  
echo "Age: $age"
```

Yakka va juft qo'shtirnoqlar

Bash quyidagilarni farqlaydi:

```
'text'
```

va:

```
"text"
```

Juft qo'shtirnoqlarda o'zgaruvchi almashtiriladi:

```
name="Ahmad"  
echo "Hello $name"
```

Natija:

```
Hello Ahmad
```

Yakka qo'shtirnoqlarda o'zgaruvchi odatda ochilmaydi:

```
echo 'Hello $name'
```

Natija:

```
Hello $name
```

Maxsus o'zgaruvchilar

Bash maxsus o'zgaruvchilarni taqdim etadi.

\$0

Skript nomi.

\$1, \$2, ...

Buyruq satri argumentlari.

\$#

Argumentlar soni.

\$?

Oxirgi buyruqning qaytish kodi.

\$\$

Joriy Shell jarayonining PID si.

\$HOME

Foydalanuvchining uy katalogi.

\$USER

Foydalanuvchi nomi.

Misol:

```
echo "Script: $0"
echo "PID: $$"
echo "User: $USER"
```

Buyruq satri argumentlari

Qaraylik:

```
#!/bin/bash

echo "Hello, $1"
```

Ishga tushirish:

```
./hello.sh Ahmad
```

Natija:

```
Hello, Ahmad
```

Bu yerda:

```
$1 = Ahmad
```

Agar quyidagini bajarsak:

```
./hello.sh Ahmad Tashkent
```

u holda:

```
$1 = Ahmad
$2 = Tashkent
```

Argumentlar sonini tekshirish

`$#` dan foydalanish mumkin.

```
#!/bin/bash

if [ "$#" -ne 1 ]; then
    echo "Usage: $0 <name>"
    exit 1
fi

echo "Hello, $1"
```

Bu yerda foydalanuvchi aynan bitta argument uzatganligi tekshiriladi.

if shartlari

Asosiy sintaksis:

```
if shart; then
    buyruqlar
elif boshqa_shart; then
    buyruqlar
else
    buyruqlar
fi
```

Misol:

```
#!/bin/bash

age=20

if [ "$age" -ge 18 ]; then
    echo "Adult"
else
    echo "Minor"
fi
```

Solishtirish operatorlari

Sonlar uchun:

- eq teng
- ne teng emas
- gt katta
- ge katta yoki teng
- lt kichik
- le kichik yoki teng

Misol:

```
if [ "$age" -ge 18 ]; then  
    echo "Adult"  
fi
```

Satrlarni solishtirish

Quyidagilar ishlatiladi:

```
=   satrlar teng
!=  satrlar teng emas
-z  satr bo'sh
-n  satr bo'sh emas
```

Misol:

```
name="Ahmad"

if [ -n "$name" ]; then
    echo "Name is not empty"
fi
```

Fayllarni tekshirish

Bash fayl xossalarini tekshirish imkonini beradi.

Asosiy operatorlar:

- e mavjud
- f oddiy fayl
- d katalog
- r o'qish uchun mavjud
- w yozish uchun mavjud
- x bajarish uchun mavjud

Misol:

```
if [ -f "data.txt" ]; then
    echo "File exists"
else
    echo "File not found"
fi
```

for sikli

Misol:

```
for name in Ahmad Ali Sara; do  
    echo "Hello, $name"  
done
```

Natija:

```
Hello, Ahmad  
Hello, Ali  
Hello, Sara
```

Fayllarni qayta ishlash uchun sikldan foydalanish mumkin:

```
for file in *.txt; do  
    echo "$file"  
done
```

while sikli

Misol:

```
counter=1

while [ "$counter" -le 5 ]; do
    echo "$counter"
    counter=$((counter + 1))
done
```

Natija:

```
1
2
3
4
5
```

until sikli

until shart yolg'on bo'lgunga qadar siklni bajaradi.

```
counter=1
```

```
until [ "$counter" -gt 5 ]; do
```

```
  echo "$counter"
```

```
  counter=$((counter + 1))
```

```
done
```

Arifmetik amallar

Bashda arifmetik ifodalar ishlatiladi.

Masalan:

```
a=10
```

```
b=5
```

```
sum=$((a + b))
```

```
difference=$((a - b))
```

```
product=$((a * b))
```

```
division=$((a / b))
```

```
echo "$sum"
```

```
echo "$difference"
```

```
echo "$product"
```

```
echo "$division"
```

Funksiyalar

Bash funksiyalarni aniqlash imkonini beradi.

```
greet() {  
    echo "Hello, $1"  
}  
  
greet "Ahmad"
```

Funksiya tugash kodini qaytarishi mumkin:

```
check_file() {  
    if [ -f "$1" ]; then  
        return 0  
    else  
        return 1  
    fi  
}
```

Foydalanish:

```
if check_file "data.txt"; then  
    echo "File exists"  
fi
```

Tugash kodi

Har bir buyruq odatda biror kod bilan tugaydi.

Qiymat:

```
0
```

odatda muvaffaqiyatli bajarilishni anglatadi.

Noldan farqli qiymat odatda xato yoki maxsus holatni anglatadi.

Masalan:

```
ls /tmp  
echo $?
```

Agar buyruq muvaffaqiyatli bajarilgan bo'lsa:

```
0
```

exit buyrug'i

Skriptni tugatish uchun quyidagi ishlatiladi:

```
exit 0
```

Masalan:

```
if [ ! -f "config.txt" ]; then  
    echo "Configuration not found"  
    exit 1  
fi
```

Bu yerda 1 xato kodi sifatida ishlatiladi.