

4. Bash

Задание 1

Скрипт мониторинга системы и управления логами

Цель: создать скрипт, который выполняет мониторинг системы и сохраняет результаты в лог-файлы.

Требования:

1. Мониторинг ресурсов:

-Используйте команды для проверки загрузки CPU, памяти, места на диске.

-Выводите результаты в лог-файл, разделяя STDOUT и STDERR.

```
# ===== ЭТАП 1: МОНИТОРИНГ РЕСУРСОВ =====
monitor_resources() {
    local timestamp=$(date '+%Y-%m-%d %H:%M:%S')

    {
        echo "=== SYSTEM MONITORING REPORT - $timestamp ==="

        # Мониторинг CPU
        echo "CPU Usage:"
        top -bn1 | head -3 | tail -1 2>/dev/null || echo "Error getting CPU info"

        # Мониторинг памяти
        echo -e "\nMemory Usage:"
        free -h 2>/dev/null || echo "Error getting memory info"

        # Мониторинг дискового пространства
        echo -e "\nDisk Usage:"
        df -h / 2>/dev/null || echo "Error getting disk info"

        # Дополнительная информация о дисках
        echo -e "\nDetailed Disk Info:"
        df -h 2>/dev/null | grep -v tmpfs || echo "Error getting detailed disk info"

        # Мониторинг нагрузки системы
        echo -e "\nSystem Load:"
        uptime 2>/dev/null || echo "Error getting uptime"

        echo -e "=== END OF REPORT ===\n"
    } >> "$LOG_FILE" 2>> "$ERROR_LOG_FILE"
}
```

2. Управление логами:

-Реализуйте ротацию логов (например, сохраняйте логи за последние 3 дня, старые удаляйте).

```
# ===== ЭТАП 2: УПРАВЛЕНИЕ ЛОГАМИ =====
create_log_dir() {
    if [ ! -d "$LOG_DIR" ]; then
        sudo mkdir -p "$LOG_DIR" 2>> "$ERROR_LOG_FILE"
        sudo chown $USER:$USER "$LOG_DIR" 2>> "$ERROR_LOG_FILE"
        echo "$(date): Created log directory: $LOG_DIR" | tee -a "$LOG_FILE"
    fi
}

rotate_logs() {
    echo "$(date): Starting log rotation..." | tee -a "$LOG_FILE"

    # Удаляем логи старше RETENTION_DAYS дней
    find "$LOG_DIR" -name "system_monitor*.log" -mtime +$RETENTION_DAYS -delete 2>> "$ERROR_LOG_FILE"
    find "$LOG_DIR" -name "system_monitor_error*.log" -mtime +$RETENTION_DAYS -delete 2>> "$ERROR_LOG_FILE"

    # Архивируем текущие логи если они больше 1MB
    if [ -f "$LOG_FILE" ] && [ $(stat -c%s "$LOG_FILE" 2>/dev/null || echo 0) -gt 1000000 ]; then
        mv "$LOG_FILE" "${LOG_FILE}.${date +%Y%m%d_%H%M%S}" 2>> "$ERROR_LOG_FILE"
    fi

    if [ -f "$ERROR_LOG_FILE" ] && [ $(stat -c%s "$ERROR_LOG_FILE" 2>/dev/null || echo 0) -gt 1000000 ]; then
        mv "$ERROR_LOG_FILE" "${ERROR_LOG_FILE}.${date +%Y%m%d_%H%M%S}" 2>> "$ERROR_LOG_FILE"
    fi
}
```

3. Сигналы и обработка ошибок:

-Добавьте обработку сигналов (SIGINT, SIGTERM), чтобы скрипт корректно завершился.

```
# ===== ЭТАП 3: СИГНАЛЫ И ОБРАБОТКА ОШИБОК =====
setup_signal_handlers() {
    trap 'cleanup SIGINT' SIGINT
    trap 'cleanup SIGTERM' SIGTERM
    trap 'handle_error' ERR
}

cleanup() {
    local signal=$1
    echo "$(date): Received signal $signal, cleaning up and exiting..." | tee -a "$LOG_FILE"
    exit 0
}

handle_error() {
    local exit_code=$?
    local command="$BASH_COMMAND"
    echo "$(date): Error in command: $command (exit code: $exit_code)" >> "$ERROR_LOG_FILE"
    return $exit_code
}
```

4. Отладка:

-Добавьте опцию для вывода отладочной информации (set -x).

```
# ===== ЭТАП 4: ОТЛАДКА =====
setup_debug() {
    if [[ "$1" == "-x" ]] || [[ "$DEBUG" == "true" ]]; then
        set -x
        echo "$(date): Debug mode enabled" | tee -a "$LOG_FILE"
    fi
}
```

Результат: Рабочий скрипт, который можно запустить и проверить.

```
#!/bin/bash

# ===== КОНФИГУРАЦИЯ =====
LOG_DIR="/var/log/system_monitor"
LOG_FILE="$LOG_DIR/system_monitor.log"
ERROR_LOG_FILE="$LOG_DIR/system_monitor_error.log"
RETENTION_DAYS=3
MONITOR_INTERVAL=60

# ===== ОСНОВНАЯ ФУНКЦИЯ =====
main() {
    setup_debug "$1"
    setup_signal_handlers
    create_log_dir
    rotate_logs

    echo "$(date): Starting system monitoring..." | tee -a "$LOG_FILE"

    while true; do
        monitor_resources
        sleep $MONITOR_INTERVAL
        rotate_logs
    done
}

# Запуск скрипта
main "$@"

uuser@uuserPC:~/git/gigaprojects/Linux4$ sudo ./system_monitor.sh
[sudo] password for uuser:
Sat 20 Sep 21:35:00 MSK 2025: Starting log rotation...
Sat 20 Sep 21:35:00 MSK 2025: Starting system monitoring...
^CSat 20 Sep 21:35:14 MSK 2025: Received signal SIGINT, cleaning up and exiting...
```

```

Sat 20 Sep 21:35:00 MSK 2025: Starting log rotation...
Sat 20 Sep 21:35:00 MSK 2025: Starting system monitoring...
=== SYSTEM MONITORING REPORT - 2025-09-20 21:35:00 ===
CPU Usage:
%Cpu(s):  0.0 us,  0.8 sy,  0.0 ni, 99.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st

Memory Usage:
          total        used        free      shared  buff/cache   available
Mem:      31Gi         7.7Gi         15Gi         194Mi        9.1Gi        23Gi
Swap:      0B           0B           0B

Disk Usage:
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb2       915G  180G  689G  21% /

Detailed Disk Info:
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb2       915G  180G  689G  21% /
efivarfs        128K   18K  106K  15% /sys/firmware/efi/efivars
/dev/sdb1       1.1G   6.2M   1.1G   1% /boot/efi
/dev/nvme0n1p3  1.9T  887G 1021G  47% /media/uuser/System

System Load:
21:35:00 up 10:03,  1 user,  load average: 0.37, 0.25, 0.21
=== END OF REPORT ===

Sat 20 Sep 21:35:14 MSK 2025: Received signal SIGINT, cleaning up and exiting...

```

Задание 2

Система резервного копирования и проверки целостности

Цель: автоматизировать процесс резервного копирования файлов и проверку целостности.

Требования:

1. Создание резервных копий:

- Напишите скрипт для инкрементального копирования файлов из заданной директории в архивы.
- Используйте tar или rsync для создания архивов.

```
#!/bin/bash

# Конфигурация
BACKUP_DIR="/home/vagrant/backup"
SOURCE_DIR="/home/vagrant/files"
LOG_FILE="/var/log/backup.log"
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
BACKUP_NAME="backup_${TIMESTAMP}.tar.gz"
MD5_FILE="$BACKUP_DIR/backup_${TIMESTAMP}.md5"

# Функция для логирования
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOG_FILE"
}

# Функция для обработки ошибок
error_exit() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - ОШИБКА: $1" >&2
    log_message "ОШИБКА: $1"
    exit 1
}

# Основная функция
main() {
    log_message "=== Запуск системы резервного копирования ==="

    # Создание резервной копии
    backup_file=$(create_backup)

    if [ -n "$backup_file" ]; then
        # Проверка целостности
        verify_backup "$backup_file"

        # Очистка старых копий
        clean_old_backups

        log_message "Резервное копирование завершено успешно"
        echo "Резервное копирование завершено успешно: $backup_file"
    fi

    log_message "=== Завершение системы резервного копирования ==="
}

# Запуск основной функции
main "$@"
```

Этап 1: Создание резервной копии

```
create_backup() {  
    log_message "Начало создания резервной копии: $BACKUP_NAME"  
  
    # Создание архива с помощью tar  
    tar -czf "$BACKUP_DIR/$BACKUP_NAME" -C "$SOURCE_DIR" . 2>/dev/null
```

2. Проверка целостности:

-После создания архива выполните проверку целостности (например, с помощью md5sum).

Этап 2: Проверка целостности архива

```
verify_backup() {  
    local backup_file=$1  
  
    log_message "Проверка целостности архива: $backup_file"  
  
    # Создание MD5 хеша  
    md5sum "$BACKUP_DIR/$backup_file" > "$MD5_FILE"  
  
    # Проверка целостности  
    if md5sum -c "$MD5_FILE" --status; then  
        log_message "Проверка целостности пройдена успешно: $backup_file"  
        return 0  
    else  
        error_exit "Ошибка проверки целостности архива: $backup_file"  
    fi  
}
```

Этап 3: Очистка старых резервных копий

```
clean_old_backups() {  
    log_message "Очистка старых резервных копий (старше 30 дней)"  
    find "$BACKUP_DIR" -name "backup_*.tar.gz" -mtime +30 -delete  
    find "$BACKUP_DIR" -name "backup_*.md5" -mtime +30 -delete  
}
```

3. Уведомления:

-Если резервное копирование прошло успешно, добавьте запись в лог-файл.

-В случае ошибки перенаправьте сообщение в STDERR.

```
if [ $? -eq 0 ]; then  
    log_message "Резервная копия успешно создана: $BACKUP_NAME"  
    echo "$BACKUP_NAME"  
else  
    error_exit "Ошибка при создании архива"  
fi  
}
```

4. Управление расписанием:

-Настройте выполнение скрипта через cron.

```
# m h dom mon dow    command
* * * * * /home/vagrant/backup_system.sh
```

~
~
~
~

"/tmp/crontab.9v50ux/crontab" 29L, 935B

Результат: полностью автоматизированная система резервного копирования.

```
2025-09-20 20:38:01 - === Запуск системы резервного копирования ===
2025-09-20 20:38:01 - Начало создания резервной копии: backup_20250920_203801.tar.gz
2025-09-20 20:38:01 - Резервная копия успешно создана: backup_20250920_203801.tar.gz
2025-09-20 20:38:01 - Проверка целостности архива: backup_20250920_203801.tar.gz
2025-09-20 20:38:01 - Проверка целостности пройдена успешно: backup_20250920_203801.tar.gz
2025-09-20 20:38:01 - Очистка старых резервных копий (старше 30 дней)
2025-09-20 20:38:01 - Резервное копирование завершено успешно
2025-09-20 20:38:01 - === Завершение системы резервного копирования ===
2025-09-20 20:39:01 - === Запуск системы резервного копирования ===
2025-09-20 20:39:01 - Начало создания резервной копии: backup_20250920_203901.tar.gz
2025-09-20 20:39:01 - Резервная копия успешно создана: backup_20250920_203901.tar.gz
2025-09-20 20:39:01 - Проверка целостности архива: backup_20250920_203901.tar.gz
2025-09-20 20:39:01 - Проверка целостности пройдена успешно: backup_20250920_203901.tar.gz
2025-09-20 20:39:01 - Очистка старых резервных копий (старше 30 дней)
2025-09-20 20:39:01 - Резервное копирование завершено успешно
2025-09-20 20:39:01 - === Завершение системы резервного копирования ===
```

~

"/var/log/backup.log" 32L, 3496B

vagrant@ubuntu-server-2:~\$ ls -l ./backup

total 88

```
-rw-rw-r-- 1 vagrant vagrant 85 Sep 20 20:29 backup_20250920_202909.md5
-rw-rw-r-- 1 vagrant vagrant 155 Sep 20 20:29 backup_20250920_202909.tar.gz
-rw-r--r-- 1 root root 85 Sep 20 20:29 backup_20250920_202915.md5
-rw-r--r-- 1 root root 155 Sep 20 20:29 backup_20250920_202915.tar.gz
-rw-rw-r-- 1 vagrant vagrant 85 Sep 20 20:34 backup_20250920_203401.md5
-rw-rw-r-- 1 vagrant vagrant 155 Sep 20 20:34 backup_20250920_203401.tar.gz
-rw-rw-r-- 1 vagrant vagrant 85 Sep 20 20:35 backup_20250920_203501.md5
-rw-rw-r-- 1 vagrant vagrant 155 Sep 20 20:35 backup_20250920_203501.tar.gz
```


Анализ процессов и диагностика системы

Цель: провести диагностику системы с помощью инструментов анализа процессов.

Требования:

1. Анализ ресурсов:

-Напишите скрипт, который запускает несколько процессов с высокой нагрузкой на CPU, память и диск.

-Используйте top, htop и /proc для анализа.

top - 13:28:14 up 5 min, 1 user, load average: 3.29, 1.09, 0.39
Tasks: 152 total, 5 running, 147 sleeping, 0 stopped, 0 zombie
%Cpu(s): 88.2 us, 11.5 sy, 0.0 ni, 0.0 id, 0.2 wa, 0.0 hi, 0.0 si, 0.2 st
MiB Mem : 1963.8 total, 276.4 free, 1211.9 used, 475.5 buff/cache
MiB Swap: 2048.0 total, 2040.1 free, 7.9 used. 588.1 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1959	vagrant	20	0	72432	3872	2264	R	60.7	0.2	0:31.17	stress-ng
1963	vagrant	20	0	1121012	1.0g	596	R	56.2	52.2	0:26.47	stress-ng
1960	vagrant	20	0	72432	3864	2264	R	48.1	0.2	0:32.02	stress-ng
1962	vagrant	20	0	72432	1140	1016	R	28.6	0.1	0:14.23	stress-ng
92	root	20	0	0	0	0	S	5.2	0.0	0:01.78	kswapd0
102	root	20	0	0	0	0	D	1.0	0.0	0:00.45	kworker/u4:3+flush-253:0
31	root	20	0	0	0	0	S	0.3	0.0	0:00.25	kcompactd0
97	root	20	0	0	0	0	I	0.3	0.0	0:00.48	kworker/u4:2-flush-253:0
1	root	20	0	166392	8944	5492	S	0.0	0.4	0:00.62	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	20	0	0	0	T	0.0	0.0	0:00.00	rcu_scp_gp

0[|||||||||||||||||||||||||||||||||||||||||100.0%]
1[|||||||||||||||||||||||||||||||||||||||||100.0%]
Mem[|||||||||||||||||||||209M/1.92G]
Swp[|6.85M/2.00G]

Tasks: 37, 41 thr; 2 running
Load average: 1.99 0.68 0.25
Uptime: 00:04:54

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
398	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.03	/sbin/multipathd -d -s
406	root	20	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
407	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
409	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
410	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
411	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
412	root	RT	0	282M	27100	9072	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
933	root	20	0	1782M	24344	4880	S	0.0	1.2	0:00.11	/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.
952	root	20	0	1782M	24344	4880	S	0.0	1.2	0:00.01	/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.
954	root	20	0	1782M	24344	4880	S	0.0	1.2	0:00.01	/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.
955	root	20	0	1782M	24344	4880	S	0.0	1.2	0:00.00	/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.
957	root	20	0	1782M	24344	4880	S	0.0	1.2	0:00.00	/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.

-Используйте ps для вывода информации о процессах.

```
vagrant@ubuntu-server-2:~$ bash -c '(sleep 10 & exec sleep 300)' &
[1] 2173
vagrant@ubuntu-server-2:~$ ps aux | grep -w Z
vagrant      2177  0.0  0.1  7008   2016 pts/1    S+   14:46   0:00 grep --color=auto -w Z
vagrant@ubuntu-server-2:~$ ps aux | grep -w Z
vagrant      2175  0.0  0.0      0      0 pts/1    Z    14:46   0:00 [sleep] <defunct>
```

3. Использование strace:

-Выполните анализ системных вызовов для команды ls с помощью strace.

```
vagrant@ubuntu-server-2:~$ strace ls
execve("/usr/bin/ls", ["ls"], 0x7ffc5d63afc0 /* 32 vars */) = 0
brk(NULL)                               = 0x560cb1de7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffa2338d90) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f99b452f000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=20864, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 20864, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f99b4529000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=166280, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 177672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f99b44fd000
mprotect(0x7f99b4503000, 139264, PROT_NONE) = 0
mmap(0x7f99b4503000, 106496, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f99b4503000
mmap(0x7f99b451d000, 28672, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) = 0x7f99b451d000
mmap(0x7f99b4525000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x27000) = 0x7f99b4525000
mmap(0x7f99b4527000, 5640, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f99b4527000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
```

-Выведите в лог-файл список вызванных системных функций.

```
vagrant@ubuntu-server-2:~$ strace -o ls_strace.log ls
backup      crashtest1.sh  files      readtest.sh      stress.log      user_management.log  users.csv
backup_system.sh  crashtest2.sh  ls_strace.log  remove_users.csv  stress_test.sh  user_manager.sh      writetest.sh
vagrant@ubuntu-server-2:~$ cat ls_strace.log
execve("/usr/bin/ls", ["ls"], 0x7ffc3caca90 /* 32 vars */) = 0
brk(NULL)                               = 0x560b870f5000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc161d9b20) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6a3f326000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=20864, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 20864, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6a3f320000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=166280, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 177672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6a3f2f4000
mprotect(0x7f6a3f2fa000, 139264, PROT_NONE) = 0
mmap(0x7f6a3f2fa000, 106496, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f6a3f2fa000
mmap(0x7f6a3f314000, 28672, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) = 0x7f6a3f314000
mmap(0x7f6a3f31c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x27000) = 0x7f6a3f31c000
mmap(0x7f6a3f31e000, 5640, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f6a3f31e000
```

4. Оптимизация:

-Измените приоритет одного из процессов (с помощью nice или ionice) и посмотрите, как это влияет на производительность.

```
top - 15:10:00 up 15 min, 2 users, load average: 0.56, 0.37, 0.16
Tasks: 141 total, 3 running, 138 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 99.8 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.2 st
MiB Mem : 1963.8 total, 1303.3 free, 211.5 used, 449.0 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1594.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1510	vagrant	39	19	72428	5952	3576	R	100.0	0.3	0:13.52	stress-ng
1509	vagrant	39	19	72428	5952	3576	R	99.7	0.3	0:13.51	stress-ng
1	root	20	0	101084	11948	8380	S	0.0	0.6	0:00.67	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

```
vagrant@ubuntu-server-2:~$ nice -n 19 stress-ng --cpu 2 --timeout 120s &
[1] 1508
vagrant@ubuntu-server-2:~$ stress-ng: info: [1508] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [1508] dispatching hogs: 2 cpu
^C
vagrant@ubuntu-server-2:~$ sudo renice -n -20 1510
1510 (process ID) old priority 19, new priority -20
vagrant@ubuntu-server-2:~$ stress-ng: info: [1508] successful run completed in 120.00s (2 mins, 0.00 secs)
```

```
top - 15:11:25 up 16 min, 2 users, load average: 1.65, 0.78, 0.32
Tasks: 141 total, 3 running, 138 sleeping, 0 stopped, 0 zombie
%Cpu(s): 50.0 us, 0.0 sy, 50.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1963.8 total, 1302.1 free, 212.6 used, 449.1 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1593.3 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1510	vagrant	0	-20	72428	5952	3576	R	100.0	0.3	1:37.60	stress-ng
1509	vagrant	39	19	72428	5952	3576	R	100.0	0.3	1:37.54	stress-ng
1	root	20	0	101084	11948	8380	S	0.0	0.6	0:00.67	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

Результат: Отчет о поведении системы под нагрузкой и рекомендации по оптимизации.

Задание 4

Автоматизация управления пользователями

Цель: создать скрипт для управления пользователями системы.

Требования:

1. Создание пользователей:

-Напишите скрипт, который создает пользователей на основе данных из CSV файла.

```
username,role,password
user1,dev,pass1
user2,admin,pass2
~
~
"users.csv" 3L, 57B
```

-Задайте каждому пользователю домашнюю директорию, сгенерируйте SSH-ключи и назначьте пароль.

```
#!/bin/bash

# Конфигурация
CSV_CREATE="users.csv"
CSV_REMOVE="remove_users.csv"
LOG_FILE="user_management.log"

# Логирование
log() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

# Создание пользователей
create_users() {
    [ ! -f "$CSV_CREATE" ] && log "ERROR: $CSV_CREATE not found" && return 1

    log "Starting user creation from $CSV_CREATE"

    while IFS=',' read -r username role password; do
        username=$(echo "$username" | tr -d ' ')
        [ -z "$username" ] && continue

        if id "$username" &>/dev/null; then
            log "User $username already exists"
            continue
        fi

        if useradd -m -s /bin/bash "$username"; then
            echo "$username:$password" | chpasswd
            mkdir -p "/home/$username/.ssh"
            chmod 700 "/home/$username/.ssh"
            chown "$username:$username" "/home/$username/.ssh"
            sudo -u "$username" ssh-keygen -t rsa -b 4096 -f "/home/$username/.ssh/id_rsa" -N "" -q
        fi
    done
}
```

2. Управление группами:

-Добавьте пользователей в группы в зависимости от их роли.

-Проверьте и обновите права доступа на директории, связанные с группами.

```
# Добавление в группу
if ! getent group "$role" &>/dev/null; then
    groupadd "$role"
fi
usermod -a -G "$role" "$username"
```

3. Логи и отчеты:

-Все изменения записывайте в лог-файл (например, созданные пользователи, ошибки).

```
        log "Created user: $username with role: $role"
    else
        log "ERROR: Failed to create user: $username"
    fi
done < <(tail -n +2 "$CSV_CREATE")

log "User creation completed"
}
```

4. Удаление пользователей:

-Добавьте функционал для удаления пользователей, указанных в другом CSV файле, с возможностью удаления их домашней директории.

```
username
user1
user2
~
~
~
"remove_users.csv" 3L, 21B
```

```
# Удаление пользователей
remove_users() {
    [ ! -f "$CSV_REMOVE" ] && log "ERROR: $CSV_REMOVE not found" && return 1

    log "Starting user removal from $CSV_REMOVE"

    while IFS=',' read -r username; do
        username=$(echo "$username" | tr -d ' ')
        [ -z "$username" ] && continue

        if id "$username" &>/dev/null; then
            if userdel -r "$username" 2>/dev/null; then
                log "Removed user: $username with home directory"
            else
                userdel "$username" && log "Removed user: $username (home kept)"
            fi
        else
            log "User $username not found"
        fi
    done <<(tail -n +2 "$CSV_REMOVE")

    log "User removal completed"
}
```

Результат: Готовый скрипт для массового управления пользователями.

```
# Основное меню
echo "User Management Script"
echo "1. Create users from CSV"
echo "2. Remove users from CSV"
echo "3. View log"
read -p "Select option (1-3): " choice

case $choice in
    1) create_users ;;
    2) remove_users ;;
    3) [ -f "$LOG_FILE" ] && cat "$LOG_FILE" || echo "No log file found" ;;
    *) echo "Invalid option" ;;
esac
```



```
vagrant@ubuntu-server-2:~$ sudo ./user_manager.sh
User Management Script
1. Create users from CSV
2. Remove users from CSV
3. View log
Select option (1-3): 2
2025-09-20 20:55:57 - Starting user removal from remove_users.csv
2025-09-20 20:55:57 - Removed user: user1 with home directory
2025-09-20 20:55:57 - Removed user: user2 with home directory
2025-09-20 20:55:57 - User removal completed
vagrant@ubuntu-server-2:~$ sudo ./user_manager.sh
User Management Script
1. Create users from CSV
2. Remove users from CSV
3. View log
Select option (1-3): 1
2025-09-20 20:56:04 - Starting user creation from users.csv
2025-09-20 20:56:05 - Created user: user1 with role: dev
2025-09-20 20:56:07 - Created user: user2 with role: admin
2025-09-20 20:56:07 - User creation completed
vagrant@ubuntu-server-2:~$ sudo ./user_manager.sh
User Management Script
1. Create users from CSV
2. Remove users from CSV
3. View log
Select option (1-3): 3
2025-09-20 20:55:26 - Starting user creation from users.csv
2025-09-20 20:55:26 - User user1 already exists
2025-09-20 20:55:26 - User user2 already exists
2025-09-20 20:55:26 - User creation completed
2025-09-20 20:55:57 - Starting user removal from remove_users.csv
2025-09-20 20:55:57 - Removed user: user1 with home directory
2025-09-20 20:55:57 - Removed user: user2 with home directory
2025-09-20 20:55:57 - User removal completed
2025-09-20 20:56:04 - Starting user creation from users.csv
2025-09-20 20:56:05 - Created user: user1 with role: dev
2025-09-20 20:56:07 - Created user: user2 with role: admin
2025-09-20 20:56:07 - User creation completed
vagrant@ubuntu-server-2:~$
```