МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э. БАУМАНА
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Сёмкин П.С., Сёмкин А.П.

Методические материалы к лабораторным работам
по дисциплине
«Операционные системы»
(кафедра СГН3)

Лабораторная работа № 7

«ОС Ubuntu. Управление процессами»

Москва
2022 г.

ОГЛАВЛЕНИЕ

1 Цель работы

Целью работы является знакомство со средствами мониторинга и управления процессами ОС Ubuntu

2 Теоретическая часть

2.1 *Подсистема управление процессами*

Подсистема управления процессами предназначена для обеспечения эффективной реализации мультипрограммного режима в операционной системе Ubuntu.

Хотя в первую очередь она отвечает **за распределение процессоров** для выполнения процессов, функциями подсистемы управления процессами также являются **передача сигналов, загрузка модулей ядра и прием прерываний.**

В состав подсистемы управления процессами входит **планировщик процессов (process scheduler)**, обеспечивающий выделение процессорного времени процессам.

2.2 *Организация процессов и потоков*

2.2.1 Структуры данных процессов и потоков

В ОС Ubuntu и процессы, и потоки называют **задачами (task**).

Изнутри они представляют собой единую структуру данных.

Каждая задача таблицы процессов представляется в виде структура **task_struct**, служащей в роли дескриптора процесса.

В структуре **task_struct** хранятся переменные и вложенные структуры, описывающие процесс.

2.2.2 Граф состояния задачи

Задача переходит в состояние **running (выполнения)** после выделения ей процессора.

При блокировке задача переходит в состояние **sleeping (спячки)**, а при остановке работы в состояние **останов (stopped)**.

Состояние **zombie (зомби)** показывает, что выполнение задачи прекратилось, однако она еще не была удалена из системы.

Задача в состоянии **dead (смерти)** может быть удалена из системы.

Состояния **active (активный)** и **expired (неактивный)** используются при планировании выполнения процесса.

При загрузке ядра запускается процесс **init**, который использует ядро для создания всех остальных задач.

2.3 *Планирование и диспетчеризация в Ubuntu*

2.3.1 Алгоритм диспетчеризации

Диспетчеризация в ОС Ubuntu основана на потоках.

Алгоритм диспетчеризации различает три класса потоков:

1. **Потоки реального времени**, обслуживаемые по алгоритму FIFO.
2. **Потоки реального времени**, обслуживаемые в порядке циклической очереди.
3. **Потоки разделения времени**

Потоки реального времени, обслуживаемые по алгоритму FIFO, имеют наивысшие приоритеты и не могут прерываться другими потоками, за исключением такого же потока реального времени FIFO, перешедшего в состояние готовности.

Потоки реального времени, обслуживаемые в порядке циклической очереди, представляют собой то же самое, что и потоки реального времени FIFO, но с тем отличием, что они могут прерываться по таймеру.

Потоки разделения времени обслуживаются в режиме пакетной обработки с выделением им заданных квантов времени.

2.3.2 Приоритет планирования

У каждого потока есть **приоритет планирования**. Значение по умолчанию равно 2**0**, но оно может быть изменено при помощи системного вызова **nice(value)**, вычитающего значение **value** из **20**. Поскольку **value** должно находиться в диапазоне от **-20** до **+19**, приоритеты всегда попадают в промежуток от **1** до **40**.(Наивысший приоритет равен **-20**)

Помимо приоритета с каждым потоком связан квант времени выполнения, то есть количество тиков таймера, в течение которых поток может выполняться. По умолчанию системные часы тикают с частотой **100 Гц**, так что каждый тик равен **10 мс**. Этот интервал в ОС Ubuntu называют «**джиффи**» (jiffy - мгновение, миг, момент).

Центральный процессор отнимается у потока при выполнении одного из следующих условий:

1. **Квант потока уменьшился до 0.**
2. **Поток блокируется на операции ввода-вывода, семафоре и т. д.**
3. **В состояние готовности перешел ранее заблокированный поток с более высокой «добродетелью».**

Так как кванты постоянно уменьшаются, рано или поздно у любого потока квант станет нулевым.

2.3.3 Очередь выполнения

После создания задачи с помощью clone, она помещается в **очередь выполнения** (run queue) процессора, содержащую ссылки на все задачи, состязающиеся за процессорное время. Очереди выполнения, напоминают многоуровневые очереди с обратной связью, позволяют присваивать задачам различные приоритеты.

Когда задача переходит в состояние блокировки либо спячки (т.е. ожидания), или же ее выполнение прекращается по какой-либо иной причине, задача удаляется из очереди выполнения.

Чтобы отличить задачи, обладающие правом на процессорное время, от задач, которые вынуждены ожидать до наступления следующего периода дискретизации, в планировщике определены два состояния: **активный (active)** и **неактивный (expired)**. При этом планировщик осуществляет диспетчеризацию только тех задач, которые находятся в активном состоянии.

2.3.4 Изменение приоритета при планировании

В планировщике Ubuntu приоритет задачи влияет на размер кванта времени и порядок выполнения задач процессором.

Во время создания каждой задаче присваивается статический приоритет **(static priority**), называемый также правильным значением (**nice value**).

Планировщик различает 40 различных уровней приоритета: от -20 до 19.

Наименьшее значение означает наибольший приоритет в алгоритме планирования (т.е. -20 - это самый высокий приоритет. который может иметь процесс).

Одной из целей планировщика Ubuntu является обеспечение высокой степени интерактивности системы.

2.3.5 Операции планирования

Планировщик удаляет задачи из процессора в том случае, когда выполнение задачи прерывается, происходит ее приоритетное вытеснение (например, по окончании выделенного кванта времени) либо при блокировании задачи.

Каждый раз при удалении задачи из процессора, планировщик вычисляет для нее следующий квант времени.

В случае блокирования задачи либо невозможности ее выполнения по иной причине, она деактивируется (**deactivate**), то есть удаляется из очереди выполнения до тех пор, пока. не будет снова готова к выполнению.

2.3.6 Многопроцессорное планирование

Поскольку планировщик процессов управляет задачами с помощью отдельных для каждого процессора очередей выполнения, задачи, как правило, являются структурно связанными с определенным процессором.

Это означает высокую вероятность отправки процесса в последующих периодах дискретизации на тот же самый процессор, что повышает быстродействие процесса, если его данные и инструкции все еще находятся в процессорном кэше.

Однако такая схема может привести к простою одного или нескольких процессоров в многопроцессорной системе даже в то время, когда система испытывает серьезную нагрузку. Во избежание подобной ситуации в случае обнаружения простоя процесса, планировщик осуществляет **балансировку загрузки (load balancing)** для переноса задач с одного процессора на другой с целью повышения эффективности использования ресурсов.

Определение загрузки процессора выполняется планировщиком на основе данных о средней длине каждой очереди выполнения в течение нескольких последних прерываний таймера, чтобы минимизировать эффект непостоянства процессорной загрузки в алгоритме балансировки.

2.3.7 Планирование реального времени

Планировщик поддерживает жесткое планирование реального времени, пытаясь минимизировать время, затрачиваемое задачей реального времени на ожидание отправки в процессор. В отличие от обычной задачи, которая, в конце концов, помещается в список неактивных (чтобы предотвратить бесконечное откладывание выполнения процессов с низким приоритетом), задача реального времени всегда помещается в активный список по истечению выделенного ей кванта времени.

Задачи реального времени всегда выполняются с более высоким приоритетом, чем обычные задачи. Поскольку планировщик всегда выделяет процессор задаче из очереди с самым высоким приоритетом активного списка (а задачи реального времени всегда находятся в активном списке), обычная задача никогда не сможет вытеснить задачу реального времени.

Чтобы предотвратить случайное либо злонамеренное использование задач реального времени, право на их создание имеют только пользователи с привилегиями **root**.

2.4 *Управление процессами*

Управление процессами ОС Ubuntu включает в себя выполнение следующих задач:

- ***просмотр запущенных процессов***
- ***просмотр информации о процессах***
- ***поиск процессов***
- ***изменение приоритета процессов***
- ***завершение процессов***
- ***ограничение памяти доступной процессу***

В Ubuntu есть очень большое количество утилит, как утилит командной строки, так и графических утилит, для решения различных задач по управлению процессами.

2.4.1 Графическая утилита «Системный монитор»

Системный монитор относится к графическим утилитам и позволяет **просматривать список запущенных процессов, завершать процессы, следить за использованием памяти, центрального процессора и файловых систем**

У программы **Системный монитор** есть три вкладки: **Процессы Ресурсы Файловые системы**

✓ Вкладка **Процессы**

Это вкладка, которую **Системный монитор** открывает в по умолчанию. Эта вкладка отображает все выполняющиеся процессы.

Отображается **имя** Процесса, **идентификатор** Процесса**, использование ЦП**, **использование памяти** и **приоритет** каждого процесса.

На основе этой информации можно выполнять операции с процессом. Контекстное меню каждого процесса позволяет:

- ***остановить процесс***
- ***продолжить остановленный процесс***
- ***завершить процесс***
- ***уничтожить процесс***
- ***изменить приоритет***
- ***просмотреть его карты распределения памяти***
- ***просмотреть открытые используемые файлы***

- ***просмотреть его подробные свойства***

✓ Вкладка **Ресурсы**

Эта вкладка отображает информацию о системных ресурсах:

- ***использование ЦП***
- ***использование памяти и историю подкачки***
- ***использование сети***

Эта вкладка может быть использована для контроля производительности системы.

✓ Вкладка **Файловые системы**

Эта вкладка отображает информацию об устройстве (устройствах) жесткого диска системы. Вкладка позволяет просмотреть информацию об устройствах и отсортировать список:

- ***имя устройства***
- ***каталог***
- ***тип***
- ***общий размер***
- ***доступная память***
- ***используемая память***

2.4.2 Утилиты командной строки для управления процессами

ОС Ubuntu, как и все ОС, содержит большой набор утилит командной строки для управления процессами

1. Утилита **ps.** Предназначена для мониторинга процессов в режиме реального времени. Показывает список всех процессов, которые выполнялись на момент запуска утилиты. Используется во многих случаях для вывода информации о выполняемых процессах. Может выполняться как в режиме командной строки, так и имеет графическую оболочку. Команда содержит множество параметров, наиболее часто используемые из которых приведены в приложении

2. Утилита **top.** Предназначена для вывода информации о процессах в реальном времени. Процессы сортируются по максимальному занимаемому

процессорному времени, но есть возможность изменить порядок сортировки. Утилита также выводит информацию о свободных системных ресурсах.

3. Утилита **nice.** Позволяет определить текущего значения приоритета **nice** по умолчанию, понижать приоритет запускаемого или выполняемого процесса. Пользователь с правами **root** может повышать приоритет команды

4. Утилита **kill.** Предназначена для принудительного завершения процесса.

5. Утилиты **jobs, fg, bg** предназначены для перевода выполнения процессов в фоновый режиме или режим переднего плана.

Синтаксис утилит управления процессами приведён в приложении.

3 Выполнение работы

3.1 *Задание*

Познакомиться с возможности утилит Ubuntu для решения задач мониторинга и управления процессами операционной системы.

Выполнить команды управления процессами в соответствие с порядком выполнения работы и объяснить полученные результаты.

3.2 *Порядок выполнения работы*

1. Войти в систему под учётной записью **user2** (Пароль **Stud-l01**).
2. Запустить программу виртуализации **Oracle VM VirtualBox.**
3. Запустить виртуальную машину **Ubuntu_XX.**
4. Войти в систему под учётной записью **admin_kaf.**

3.2.1 Мониторинг и управление процессами с помощью графической утилиты «Системный монитор»

1. Запустить графическую утилиту **системный монитор.**
2. Отобразить информацию о выполняющиеся процессах и выполнить операции с процессами:

- используя вкладку **Процессы** отобразить информацию о процессах системы.
- запустить программу **Терминал**
- с помощью контекстного меню просмотреть информацию о процессе(**gnome-terminal-server**) и изменить приоритет процесса до **Высокий**
- завершить процесс **gnome-terminal-server**

3. Отобразить информации о ресурсах
   - используя вкладку **Ресурсы** отобразить информацию о системных ресурсах:
     Использование ЦП
     Использование памяти и подкачки
     Использование сети
   - Объяснить полученные результаты
4. Отобразить информации о файловых системах
   - используя вкладку **Файловые системы** отобразить информацию об устройствах и файловых системах
   - Объяснить полученные результаты

3.2.2 Мониторинг и управление процессами с помощью утилит командной строки

1. Открыть окна интерпретатора команд
2. Получить общую информацию о системе
   - Вывести информацию о текущем интерпретаторе команд
   - Вывести информацию о текущем пользователе
   - Вывести информацию о текущем каталоге
   - Вывести информацию об оперативной памяти и области подкачки
   - Вывести информацию о дисковой памяти
3. Получить информации о процессах

- Получить идентификатор текущего процесса(PID)
- Получить идентификатор родительского процесса(PPID)
- Получить идентификатор процесса по его имени(**init**)
- Получить информацию о выполняющихся процессах с помощью команды **ps (**параметры команды даны в приложении)

4. Выполнить команды управления процессами
   - Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе
   - Определить текущее значение приоритета по умолчанию
   - Запустить командный процессор **bash** с понижением приоритета
   - Определить **PID** запущенного командного процессора
   - Установить приоритет командного процессора равным **5**
   - Получить информацию об активных процессах пользователя, включая их приоритеты, используя утилиту **top**

4 Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu.
2. Как создаются задачи задачи в ОС Ubuntu?
3. Назовите классы потоков ОС Ubuntu
4. Как используется приоритет планирования при запуске задачи
5. Как можно изменить приоритет для выполняющейся задачи?

5 Литература

1. Робачевский А.М. Операционная система UNIX.-СПб.: БХВ-Петербург, 2001. – 528 с.:ил.
2. Негус К. Ubuntu и Debian Linux для продвинутых. 2-е изд. – СПб.: Питер,2014. -384 с.: ил.

6 Приложение

6.1 *Команды получения информации о системе*

$ echo $SHELL - получение информации о текущем интерпретаторе

$ whoami - получение информации о текущем пользователе

$ pwd - получение информации о текущем каталоге

$ free - получение информации об оперативной памяти и файле подкачки

$ df - получение информации о дисковой памяти

6.2 *Команды получения информации о процессах*

$ echo $$ - -получение идентификатора текущего процесса**(PID)**

$ echo $PPID - получение идентификатора родительского процесса**(PPID)**

$ pidof <имя процесса> - получение идентификатора процесса по его имени

$ ps <параметр> - получение информации о выполняемых процессах

Параметр	**Описание**
Без параметра	информации о выполняемых процессах текущего пользователя в текущем интерпретаторе команд
-a	отобразить все процессы, связанных с терминалом (отображаются процессы всех пользователей)
-e	отобразить все процессы
-t список терминалов	отобразить процессы, связанные с терминалами
-u идентификаторы пользователей	отобразить процессы, связанные с данными идентификаторами
-g идентификаторы групп	отобразить процессы, связанные с данными идентификаторами групп
-x	отобразить все процессы, не связанные с терминалом
lax	отобразить информацию о значения nice процессов
ejH	отобразить иерархию процессов в виде дерева

$ top <параметры> -отображение информации об активных процессах

6.3 *Команды управления процессами*

6.3.1 Изменение приоритета

$ nice - определение текущего значения nice по умолчанию

$ nice –n [коэффициент понижения] команда [аргумент] **-** понижение приоритета запускаемого процесса. Команда **nice** выполняет команду с пониженным приоритетом, коэффициент понижения указывается в диапазоне 1..19 (по умолчанию он равен 10)

$ renice –n [значение nice] **–p** [PID процесса] **-** понижение приоритета выполняемого процесса

Пользователь с правами **root** может повышать приоритет команды, для этого необходимо указать отрицательный коэффициент, например –12

6.3.2 Принудительное завершение процесса

$ kill [-номер сигнала] PID

где **PID** - идентификатор процесса, который можно узнать с помощью команды **ps**.

6.3.3 Выполнение процессов в фоновом режиме

$ jobs - выводит список процессов, которые выполняются в фоновом режиме

$ fg - переводит процесс в нормальные режим ("на передний план" - foreground)

$ bg -переводит процесс в фоновый режим.