Лабораторная работа №3

1. pidin

Синтаксис команды - pidin [options] argument

Основные аргументы:

arg: Отображает аргументы процессов.

backtrace: Показать трассировку стеков для потоков.

channels: Информация о длинах очередей отправки, получения и ответов.

env: Показать переменные окружения процессов.

extsched: Информация о конфигурации расширенного планировщика.

fam: Отображает семейные связи процессов (сессии, группы процессов, родители, дети и братья).

fds: Информация о соединениях и файловых дескрипторах процессов.

flags: Отображает флаги процессов.

info: Общая информация о системе, такая как тип процессора и объем свободной памяти.

irqs: Отображает обработчики прерываний, принадлежащие процессу.

map: Информация о отображении памяти.

mem: Показать память, используемую процессами.

net: Информация о всех узлах в сети Qnet.

pmem: Показать только память процесса.

rc: Показать имя процесса и аргументы для всех удалённых узлов.

regs: Показать значения регистров.

rmasks: Отображение масок запуска и наследования.

sched: Отображает параметры планирования потоков.

sig: Информация о состояниях сигналов процессов.

sess: Отображает информацию по сессии процесса.

threads: Отображает информацию о потоках.

timers: Показать таймеры, принадлежащие процессу.

times: Время запуска процесса, пользовательское и системное время.

ttimes: Время работы потоков.

user: Отображает идентификаторы пользователей и групп.

1. slay

slay [options]… process\_name | process\_ID

Основные ключи

-#: Использует сигнал с указанным номером.

-f: Принудительное выполнение действия, даже если найдены несколько совпадений.

-h: Послать сигнал SIGSTOP, чтобы остановить процесс.

-m <name|pid>: Поиск процесса только по имени или PID (по умолчанию ищет оба).

-n <node>: Ищет процессы на удаленном узле node.

-P <prio>: Изменяет приоритет на prio. При необходимости можно указать политику планирования (f, r, o).

-p: Выводит ID процессов в десятичном формате на stdout, по одному процессу на строку. Другие действия не выполняются, процессы не будут убиты.

-Q: Тихий режим (без вывода).

-q: Запрос перед сигналом (перекрывает -f).

-S: Не убивать процессы, если у них есть потомки.

-s <sig>: Определяет сигнал, который нужно отправить.

-t <ttyname>: Процесс должен быть на терминале ttyname.

-u: Послать сигнал SIGCONT, чтобы возобновить выполнение процесса.

-v: Подробный вывод (verbose).

-T <thread\_id>: Указывает идентификатор потока.

-R <runmask>: Устанавливает привязку процессора к runmask.

-C <cpunum>: Устанавливает привязку к процессору cpunum.

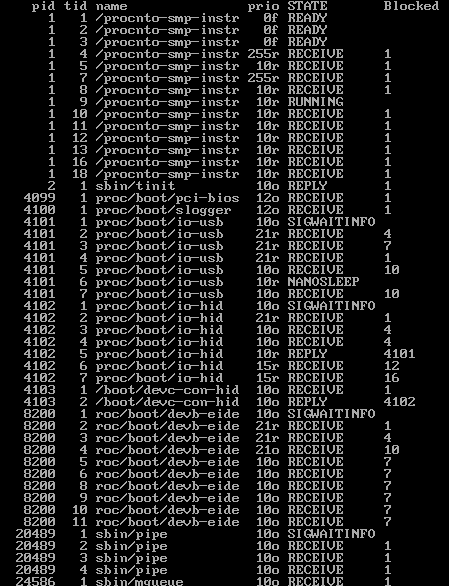
-i: Используется вместе с -R, чтобы также установить маску наследования.

3.2. Ответьте на вопросы:

3.1. tid (Thread ID) - уникальный идентификатор потока в системе.

3.2. pid (Process ID) - уникальный идентификатор процесса в системе.

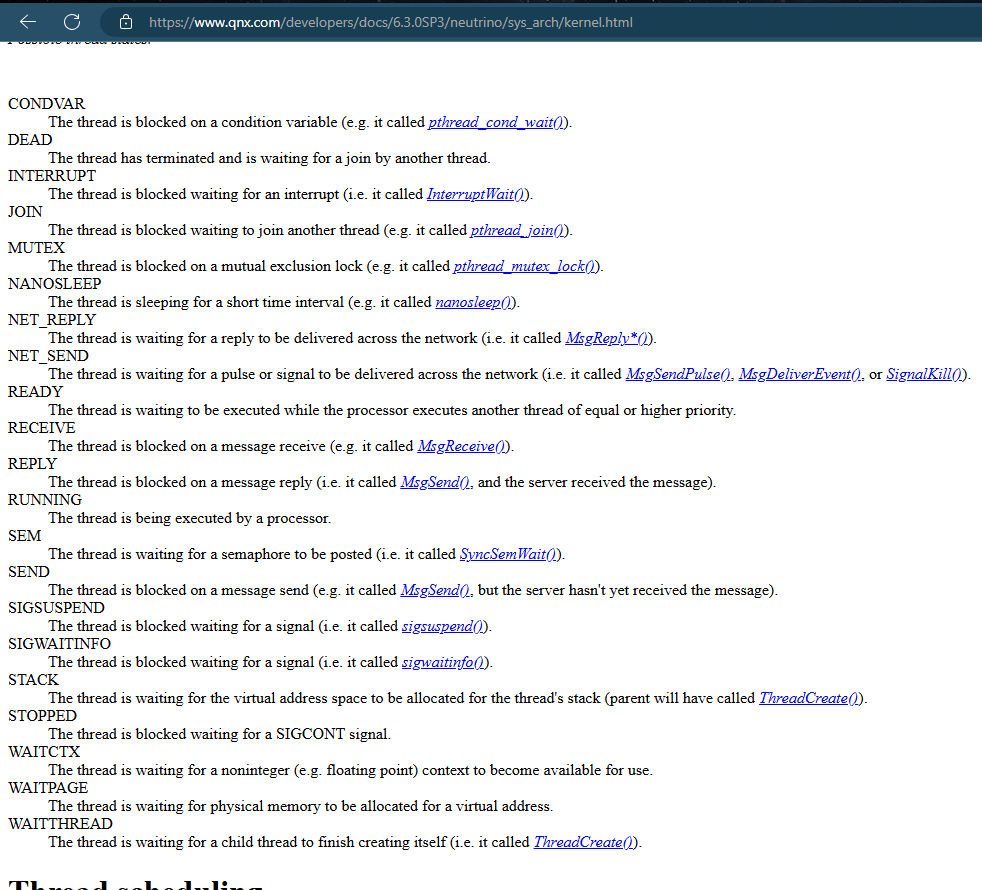
3.3. Вводим команду pidin



Первый поток имеет pid 1 и tid 1, а второй поток имеет pid 4102 и tid 3.

3.4

Данные взяты из официальной документации



CONDVAR

Поток заблокирован на условной переменной.

INTR (INTERRUPT)

Поток заблокирован в ожидании прерывания.

NANOSLEEP

Поток спит в течение короткого промежутка времени.

READY

Поток готов к выполнению, но процессор выполняет другой поток с равным или более высоким приоритетом.

RECEIVE

Поток заблокирован в ожидании получения сообщения.

REPLY

Поток заблокирован в ожидании отправки ответа на сообщение.

RUNNING

Поток выполняется на процессоре.

SIGSUSPEND

Поток заблокирован в ожидании сигнала.

SIGWAITINFO

Поток заблокирован в ожидании информации о сигнале.

3.5

CONDVAR

{84136016-3083-424E-B0F6-15FE51D5A5B0}

INTR

{76282D78-C05B-4C0A-B8F9-DC62F5303C0D}

NANOSLEEP

{B764CA24-B1B5-4570-B885-57537670E030}

READY

{48542D99-E6C3-465B-8102-16A278BD4089}

RECEIVE

{C3F2713F-346B-4179-8707-1E42A3BBF78E}

REPLY

{D35757E7-1940-4D27-B097-807B4E60B7A4}

RUNNING

{DABAE9F9-398E-44F6-8008-0958A6251472}

SIGSUSPEND

{42FC1378-5C4B-4F6F-99F3-092C942F46A4}

SIGWAITINFO

{4CC6CF30-7C07-4400-A543-759EDFB41489}

3.6. Какие дисциплины диспетчеризации потоков существуют в ОСРВ QNX? В чем суть каждой из дисциплин? В чем основные отличия этих дисциплин? Как утилита pidin отмечает разные дисциплины диспетчеризации?

В QNX существуют следующие дисциплины диспетчеризации:

1)FIFO (First In First Out — "первый вошел — первый вышел").

Если потоку назначена дисциплина диспетчеризации FIFO, то другой поток с таким же приоритетом получит управление только если исполняющийся поток заблокируется или сам уступит право исполнения.

2)"Карусельная" (Round Robin) диспетчеризация — поток исполняется в течение квантов времени и передает управление следующему потоку с таким же приоритетом. В QNX 6.3 квант времени (timeslice) по умолчанию равен 4 мс (для процессоров с частотой выше 40 МГц).

3)Спорадическая диспетчеризация — предназначена для установления лимита использования потоком процессора в течение определенного периода времени. Этот механизм заменил имевшуюся в прежних версиях QNX адаптивную диспетчеризацию.

FIFO (First In First Out):

1)Потоки выполняются в порядке их поступления.

2)Поток с таким же приоритетом получит управление только если текущий поток заблокируется или добровольно уступит управление.

Карусельная (Round Robin):

1)Потоки выполняются в течение определенного кванта времени, после чего управление передается следующему потоку с таким же приоритетом.

2)Квант времени (timeslice) по умолчанию равен 4 мс (для процессоров с частотой выше 40 МГц).

Спорадическая диспетчеризация:

1)Ограничивает использование процессора потоком в течение определенного периода времени.

2)Предназначена для контроля использования ресурсов и заменила адаптивную диспетчеризацию, которая использовалась в более ранних версиях QNX.

Утилита pidin отмечает разные дисциплины диспетчеризации с помощью букв в поле prio команды pidin sched, где f - FIFO,r - Round Robin, o - other( все остальные).

1. FIFO

{AE82BB07-CD84-46D4-92A2-03D41BB0DF5F}

1. Round Robin

{BD3A1808-C0E9-4F29-9C0C-A259ED80FF46}

1. Other

{27030A41-1745-4880-8458-FB242D2996A2}

3.8. Что такое приоритет потока? Какие приоритеты существуют? Как утилита pidin отмечает разные приоритеты потоков?

Приоритет потока — это значение, определяющее порядок выполнения потоков в системе. Планировщик выбирает следующий поток для выполнения, основываясь на приоритете: поток с самым высоким приоритетом среди готовых к выполнению (READY) выбирается для выполнения.

Какие приоритеты существуют?

Операционная система QNX поддерживает 256 уровней приоритетов потоков:

0: Специальный поток "idle" (простой), который всегда готов к выполнению и имеет наименьший приоритет.

1-63: Приоритеты, доступные для потоков, которые не запускаются от имени root (не-root потоки). Потоки могут иметь приоритет от 1 (минимальный) до 63 (максимальный для non-root).

64-255: Приоритеты, которые могут назначаться только потокам, запущенным от имени root.

Потоки с одинаковым приоритетом выполняются согласно их дисциплине диспетчеризации (например, FIFO, Round Robin).

Как утилита pidin отмечает разные приоритеты потоков?

Утилита pidin отображает приоритеты потоков в столбце PRI. Например, вы можете увидеть значения, такие как 10r, где 10 — это значение приоритета потока. Числа будут находиться в диапазоне 0–255 в зависимости от конкретного потока и его прав доступа (root или non-root).

Ссылочка на документацию:

[qnx.com/developers/docs/6.3.0SP3/neutrino/sys\_arch/kernel.html#SCHEDULING](https://www.qnx.com/developers/docs/6.3.0SP3/neutrino/sys_arch/kernel.html" \l "SCHEDULING)

3.9.Потоки с разными приоритетами

{78939F74-3E41-469E-82AB-B9AF402B1554}

{21622829-B694-4653-BEC9-DA3573F5CB53}

{7BEF7BAA-B930-4C25-A0CD-E016C6F6098B}

3.10. 5 пар разных потоков, принадлежащих разным процессам, но с одинаковым приоритетом и одинаковой ДД

{64E65D57-C511-4D0E-9C2F-CA62F25412DA}{64E65D57-C511-4D0E-9C2F-CA62F25412DA}

{D369A0E0-AFC5-48CD-B971-9B80BDA392DF}

{B5B7CB27-6E98-44F5-92D1-DD25A411A967}

{4F71A6AA-349F-4BB0-93FA-428DFC78097B}

{B041A9B4-94F4-4152-BC90-DD11A5E34301}

1. Используем slay
   1. Меняем только приоритет:

slay -P 200 8200(slay -P <приоритет> <PID>)

{216D2EE4-2923-4591-A96B-1ADF5C95CA52}

{8F41623A-6575-40A5-BF9D-8D9D5DB9C861}

* 1. Меняем только дисциплину диспетчеризации

{8F41623A-6575-40A5-BF9D-8D9D5DB9C861}

{D917BD97-DE1F-4517-8A7F-D6B856D7BC02}

4.3. Меняем и дисциплину диспетчеризации и приоритет

{D917BD97-DE1F-4517-8A7F-D6B856D7BC02}

{7F909FEB-3040-471C-ACD3-9480ABC573DE}