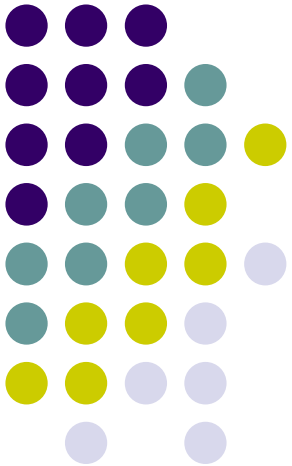
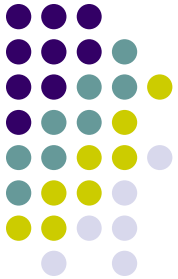
Операционные системы

Процессы

Управление процессами



Функции ОС

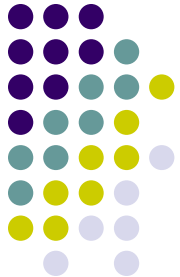


- Управление памятью
- Управление процессами
- Управление оборудованием
- Интерфейс

Процесс

- Что такое «процесс»?
  - Программа?
    - Программа — это статическая последовательность команд
  - Исполняющаяся программа?
    - В моменты когда процесс не исполняется на процессоре?
  - Заявка на потребление системных ресурсов
  - Абстракция, описывающая выполняющуюся программу

Понятие процесса

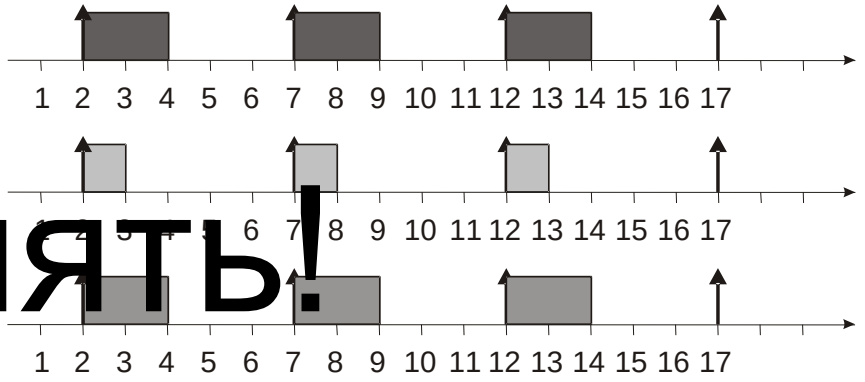


- Совокупность
  - набора исполняющихся команд
  - ассоциированных с ним ресурсов
    - адресное пространство, стеки, файлы, устройства ввода-вывода и т.д.
  - и данных состояния исполнения
    - значения регистров, программного счетчика, состояние стека и значения переменных
- находящаяся под управлением операционной системы

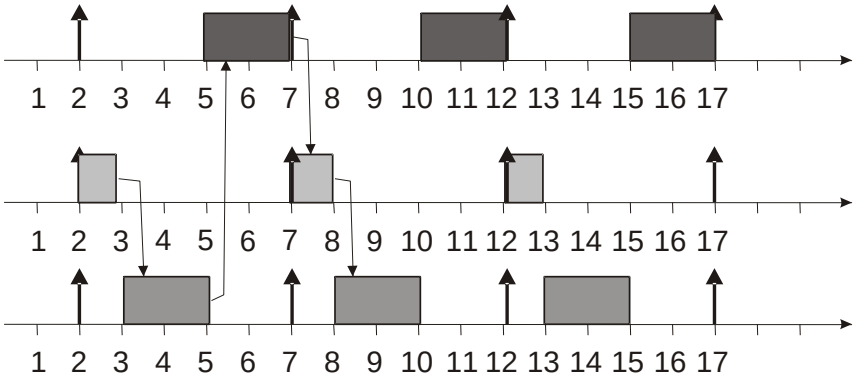
Что делать с процессами?

- Если количество задач ≤ числа процессоров в системе, то результат очевиден

Управлять!

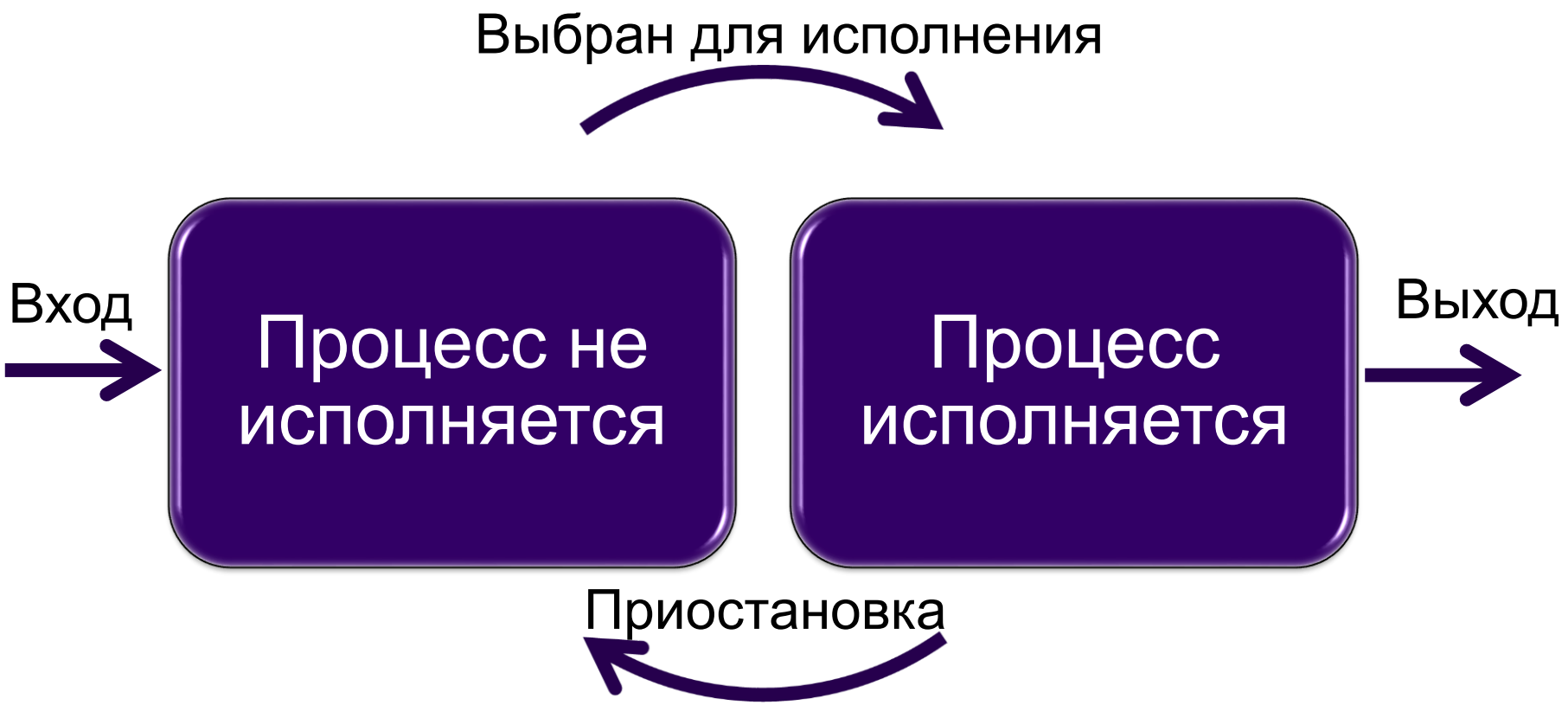


- В случае одного процессора время приходится делить

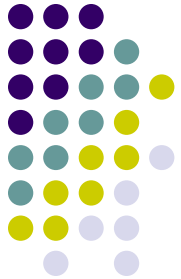


Состояния процесса

- В простейшем состоянии

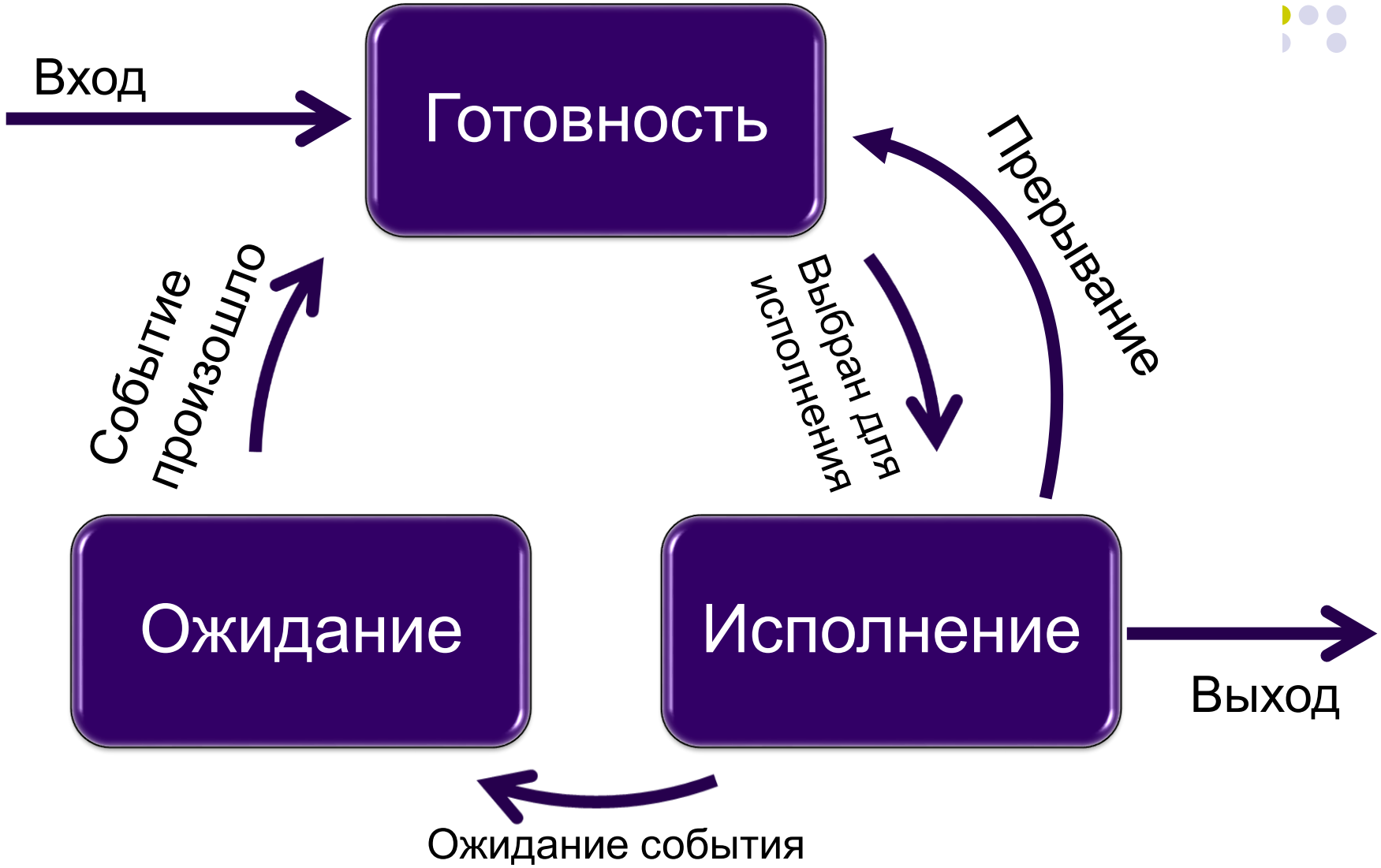


События и состояние процесса

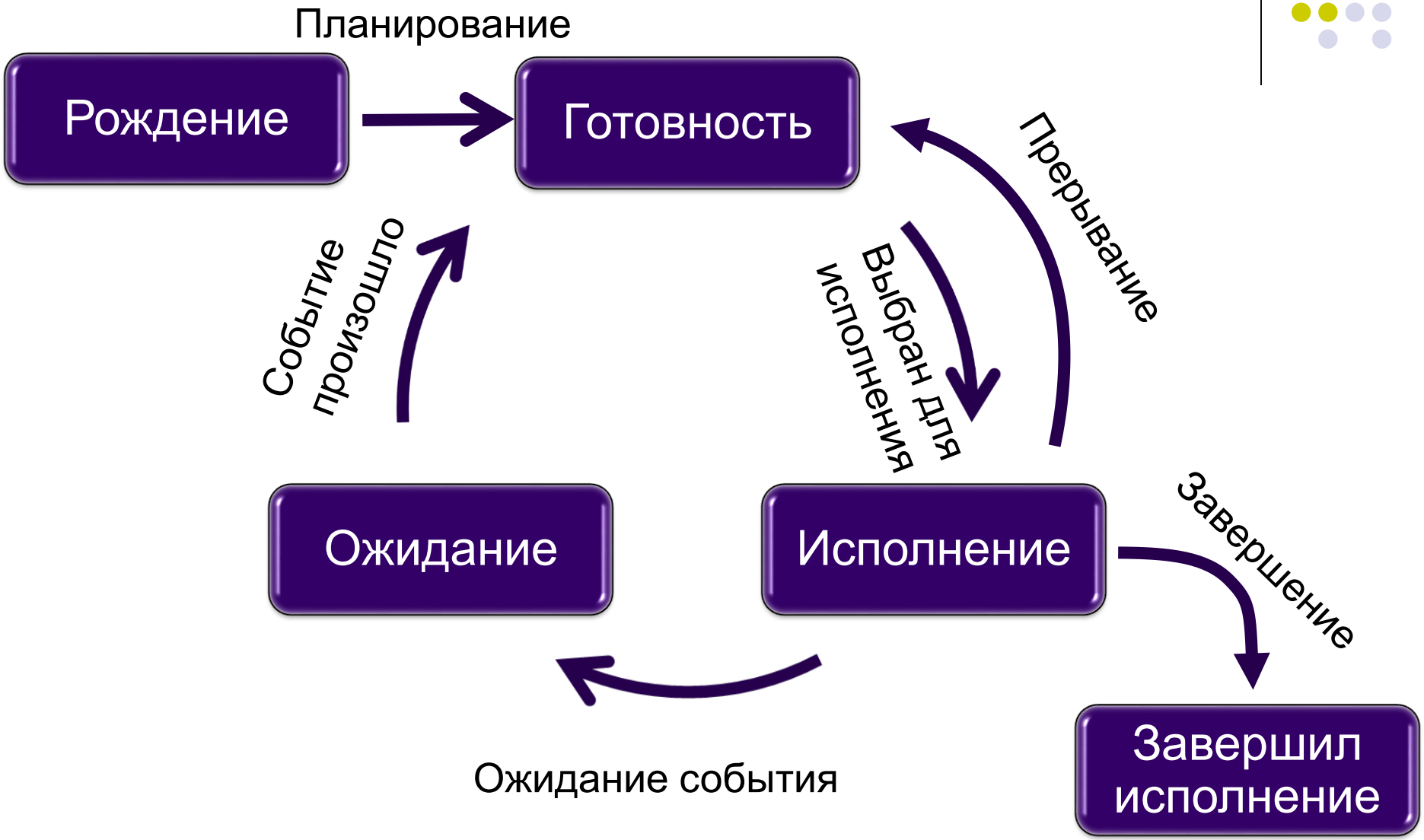


- Процесс может ожидать некоторых событий/ресурсов
  - «Неисполнение» разделяется на 2
- Состояния процесса
  - Исполнение
  - Ожидание
  - Готовность

События и состояние процесса



Состояния процесса



Операции над процессом

- Изменением состояния процессов занимается операционная система
  - Создание процесса — завершение процесса
  - Приостановка — запуск
  - Блокирование — разблокирование

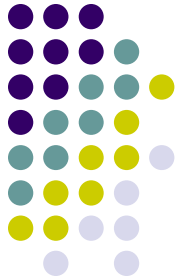
Блок управления процессом

- Process Control Block — PCB
- Структура данных однозначно определяющая состояние процесса в ОС
- При выполнении операций с процессом, ОС модифицирует PCB процесса
- В каждой ОС своя структура PCB
  - Несколько структур
  - Разные наборы данных

Блок управления процессом

- Состояние, в котором находится процесс
- Программный счетчик процесса
- Содержимое регистров процессора
- Данные, необходимые для планирования
- Учетные данные
- Информацию об устройствах ввода-вывода, связанных с процессом

PCB

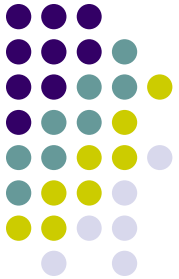


- Программный счетчик процесса
  - Адрес команды, которая должна быть выполнена для него следующей (IP)
- Данные, необходимые для планирования использования процессора и управления памятью
  - Приоритет процесса, размер и расположение адресного пространства и т. д.
- Учетные данные
  - Идентификационный номер процесса, PID
  - Пользователь-владелец процесса, время использования процессора и т. д.
- Устройства ввода-вывода
  - Какие устройства закреплены за процессом, таблица открытых файлов

Контекст процесса

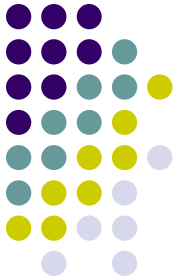
- Весь PCB рассматривают по частям
  - Регистровый контекст
    - Значения всех регистров, в том числе IP и не доступных коду процесса (LDTR, TR, …)
  - Системный контекст
    - Все данные необходимые ОС для поддержания процесса
  - Пользовательский контекст
    - Код и данные процесса

Создание процесса



- Сложный жизненный путь ~~человека~~… процесса в компьютере начинается с его рождения

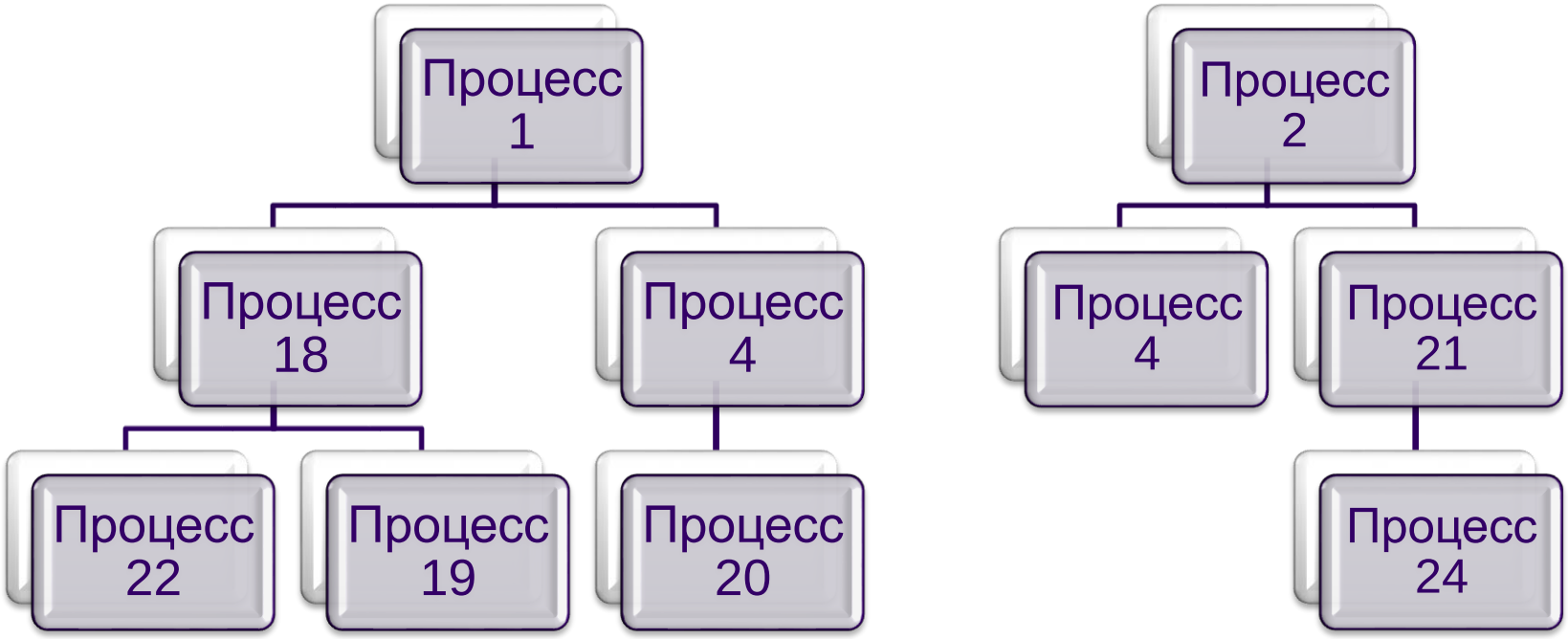
Создание процесса



- Любая ОС обеспечивает создание процессов
  - При запуске
  - Динамически
    - Процесс пользователя
      - Системный вызов
    - ОС
      - Тоже процесс
    - Родительский процесс
      - Дочерний процесс

Лес процессов

- Генеалогический лес деревьев



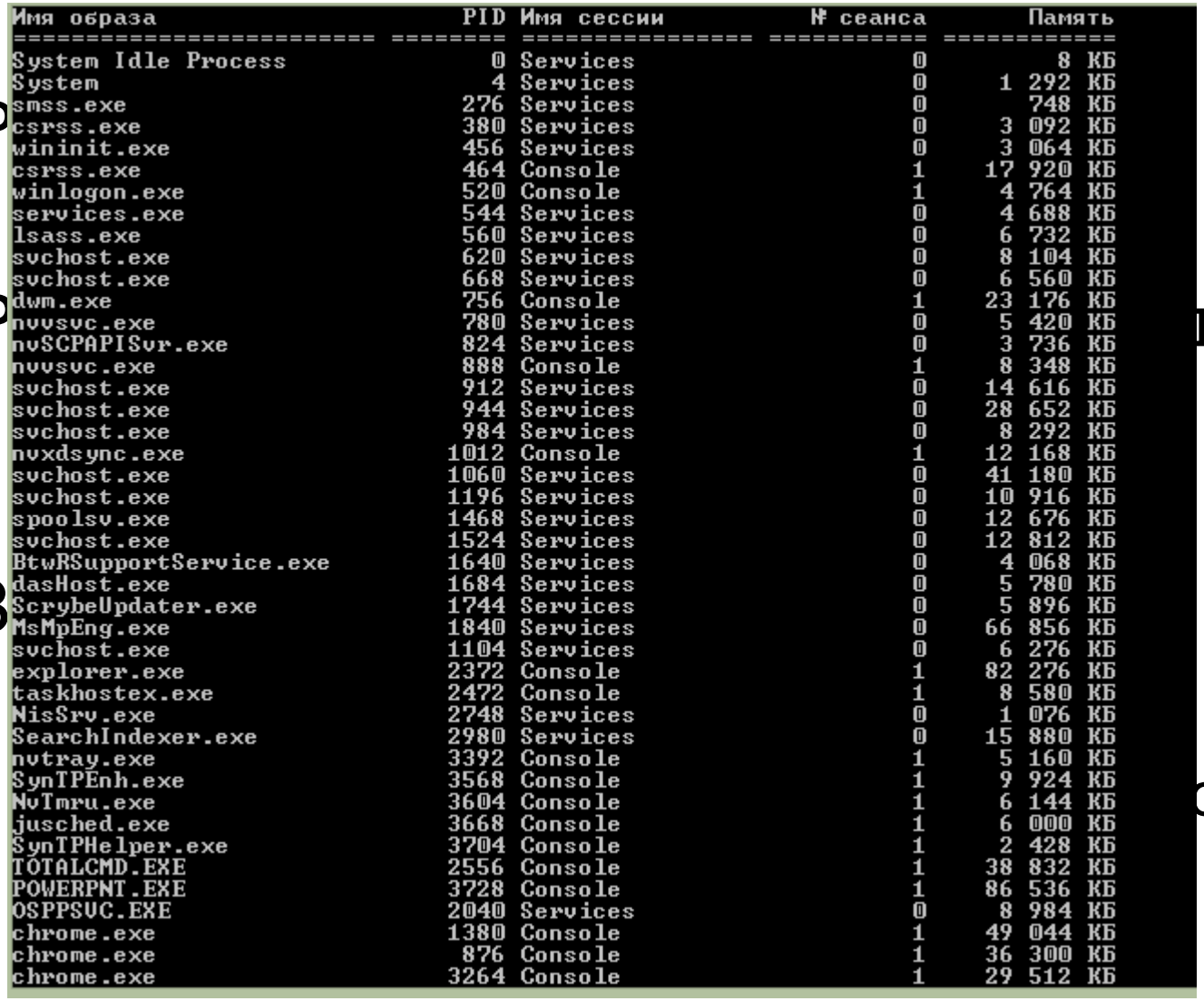
В некоторых системах лес вообще вырождается в одно дерево

Дерево процессов

Process	PID	CPU	Description	Company Name
System Idle Process	0	98.83		
Interrupts	n/a	0.77	Hardware Interrupts	
DPCs	n/a		Deferred Procedure Calls	
System	4			
smss.exe	276			
csrss.exe	380			
wininit.exe	456			
services.exe	544			
lsass.exe	560			
csrss.exe	464			
winlogon.exe	520			
dwm.exe	756			
explorer.exe	2372		Проводник	Microsoft Corporation
SynTPEnh.exe	3568		Synaptics TouchPad Enhan...	Synaptics Incorporated
SynTPHelper.exe	3704			
NvTmru.exe	3604		NVIDIA NvTmru Application	NVIDIA Corporation
jusched.exe	3668		Java(TM) Update Scheduler	Oracle Corporation
TOTALCMD.EXE	2556		Total Commander 32 bit	Ghisler Software GmbH
POWERPNT.EXE	3728		Microsoft PowerPoint	Microsoft Corporation
WINWORD.EXE	4040		Microsoft Word	Microsoft Corporation
splwow64.exe	2316		Print driver host for applicatio...	Microsoft Corporation
AIMP3.exe	3512		AIMP3	AIMP DevTeam
cmd.exe	1872		Обработчик команд Windo...	Microsoft Corporation
conhost.exe	612		Окно консоли узла	Microsoft Corporation
bash.exe	3600			
procexp.exe	2456	0.77	Sysinternals Process Explorer	Sysinternals - www.sysinter...
chrome.exe	1380		Google Chrome	Google Inc.
chrome.exe	876		Google Chrome	Google Inc.
chrome.exe	3264		Google Chrome	Google Inc.
chrome.exe	3004		Google Chrome	Google Inc.
chrome.exe	1648		Google Chrome	Google Inc.

Создание процессов

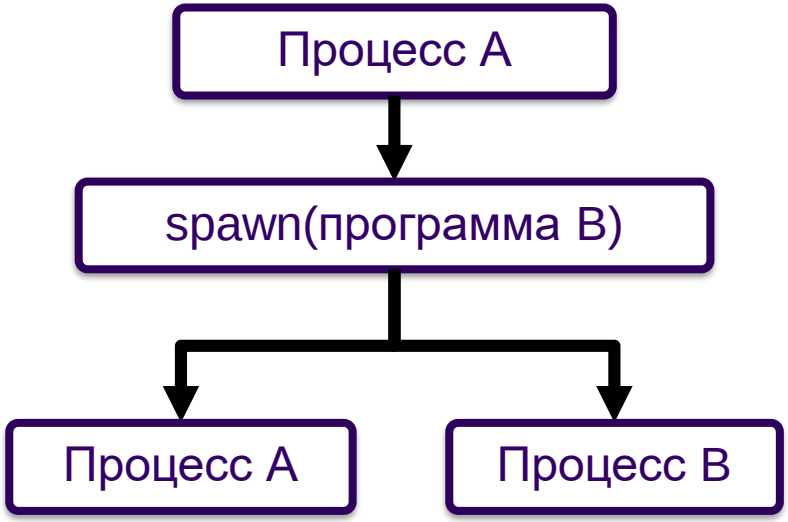
- P
  -
- P
  -
  -
  -
- З
  -
  -
  -

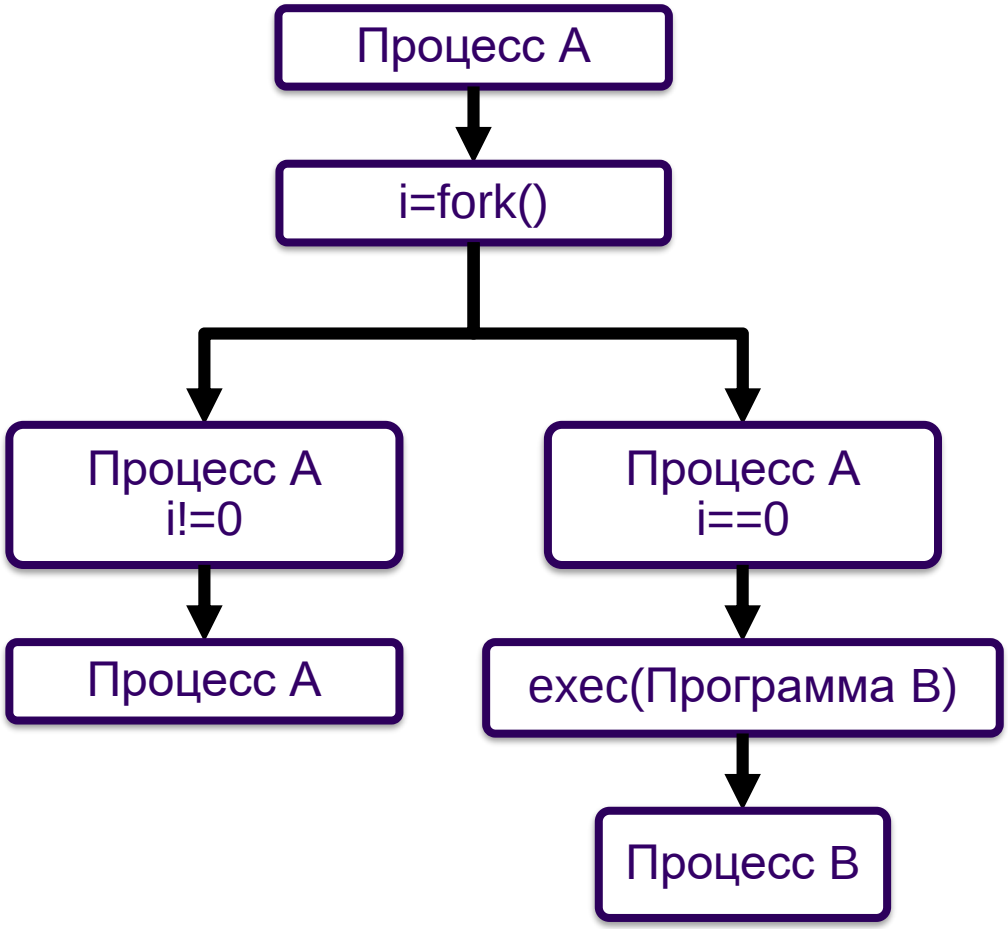


Имя образа	PID	Имя сессии	№ сеанса	Память
System Idle Process	0	Services	0	8 КБ
System	4	Services	0	1 292 КБ
smss.exe	276	Services	0	748 КБ
csrss.exe	380	Services	0	3 092 КБ
wininit.exe	456	Services	0	3 064 КБ
csrss.exe	464	Console	1	17 920 КБ
winlogon.exe	520	Console	1	4 764 КБ
services.exe	544	Services	0	4 688 КБ
lsass.exe	560	Services	0	6 732 КБ
svchost.exe	620	Services	0	8 104 КБ
svchost.exe	668	Services	0	6 560 КБ
dwm.exe	756	Console	1	23 176 КБ
nvvsvc.exe	780	Services	0	5 420 КБ
nvSCPAPISvr.exe	824	Services	0	3 736 КБ
nvvsvc.exe	888	Console	1	8 348 КБ
svchost.exe	912	Services	0	14 616 КБ
svchost.exe	944	Services	0	28 652 КБ
svchost.exe	984	Services	0	8 292 КБ
nvxdsync.exe	1012	Console	1	12 168 КБ
svchost.exe	1060	Services	0	41 180 КБ
svchost.exe	1196	Services	0	10 916 КБ
spoolsv.exe	1468	Services	0	12 676 КБ
svchost.exe	1524	Services	0	12 812 КБ
BtwRSupportService.exe	1640	Services	0	4 068 КБ
dasHost.exe	1684	Services	0	5 780 КБ
ScrybeUpdater.exe	1744	Services	0	5 896 КБ
MsMpEng.exe	1840	Services	0	66 856 КБ
svchost.exe	1104	Services	0	6 276 КБ
explorer.exe	2372	Console	1	82 276 КБ
taskhostex.exe	2472	Console	1	8 580 КБ
NisSrv.exe	2748	Services	0	1 076 КБ
SearchIndexer.exe	2980	Services	0	15 880 КБ
nvtray.exe	3392	Console	1	5 160 КБ
SynTPEnh.exe	3568	Console	1	9 924 КБ
NvTmru.exe	3604	Console	1	6 144 КБ
jusched.exe	3668	Console	1	6 000 КБ
SynTPHelper.exe	3704	Console	1	2 428 КБ
TOTALCMD.EXE	2556	Console	1	38 832 КБ
POWERPNT.EXE	3728	Console	1	86 536 КБ
OSPPSVC.EXE	2040	Services	0	8 984 КБ
chrome.exe	1380	Console	1	49 044 КБ
chrome.exe	876	Console	1	36 300 КБ
chrome.exe	3264	Console	1	29 512 КБ

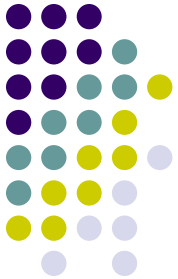
Порождение процессов Linux

- Использует первый тип создания процессов
  - Системные вызовы
    - fork+exec
    - spawn





Особенности fork+exec и spawn



- fork
  - Порожденный процесс наследует все ресурсы родителя!
  - Порожденный процесс начинает выполняться с того места, где был выполнен вызов
  - Перед вызовом exec порожденный процесс может создать новые ресурсы
- exec
  - Порожденный процесс наследует часть ресурсов родителя
  - Порожденный процесс начинает выполняться с начала

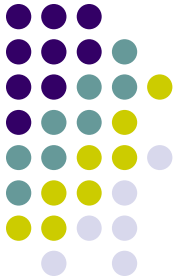
- spawn
  - Порожденный процесс наследует часть ресурсов родителя
  - Порожденный процесс начинает выполняться с начала

Fork-бомбы

```
while (1) {
        fork();
}
```

- Приводит к мгновенному заполнению таблицы дескрипторов и экспоненциальному росту накладных расходов на поддержание всех процессов
  - Проблема 65535 процессов

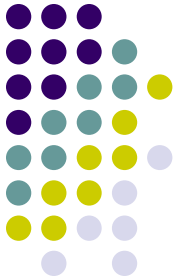
Создание процессов Windows



- Использует только второе решение
- Системные вызовы
  - Создают новый набор ресурсов
  - Заполняют сегмент кодов программой

- WINBASEAPI UINT WINAPI **WinExec**(
  __in LPCSTR lpCmdLine,
  __in UINT uCmdShow
  );

Создание процессов Windows



- WINBASEAPI BOOL WINAPI **CreateProcessW**(
  __in_opt LPCWSTR lpApplicationName,
  __inout_opt LPWSTR lpCommandLine,
  __in_opt LPSECURITY_ATTRIBUTES lpProcessAttributes,
  __in_opt LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in BOOL bInheritHandles,
  __in DWORD dwCreationFlags,
  __in_opt LPVOID lpEnvironment,
  __in_opt LPCWSTR lpCurrentDirectory,
  __in LPSTARTUPINFOW lpStartupInfo,
  __out LPPROCESS_INFORMATION lpProcessInformation
  );

Порождение программ

- Дублирование процессов
  - Параллельные программы
  - Несколько процессов для одной программы

- Замена пользовательского контекста
  - В рамках **одного процесса** могут последовательно исполнится **несколько различных программ**

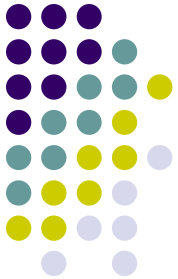
Завершение процессов

- Может происходить по различным причинам:
  - Естественные — от старости
  - Принудительно — убийство
    - ОС просто не выделяет процессу времени и …
- ОС переводит процесс в состояние **закончил исполнение**
  - И освобождает все ресурсы, память, исключает из планирования
  - Оставляя только PCB

PCB завершенного процесса

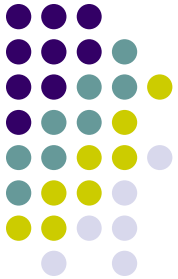
- Сохраняется
  - Для того чтобы родительский процесс мог получить информацию о завершении дочернего
  - Код завершения
    - exit(7)
  - До тех пор пока существует родительский процесс

PCB завершенного процесса Windows



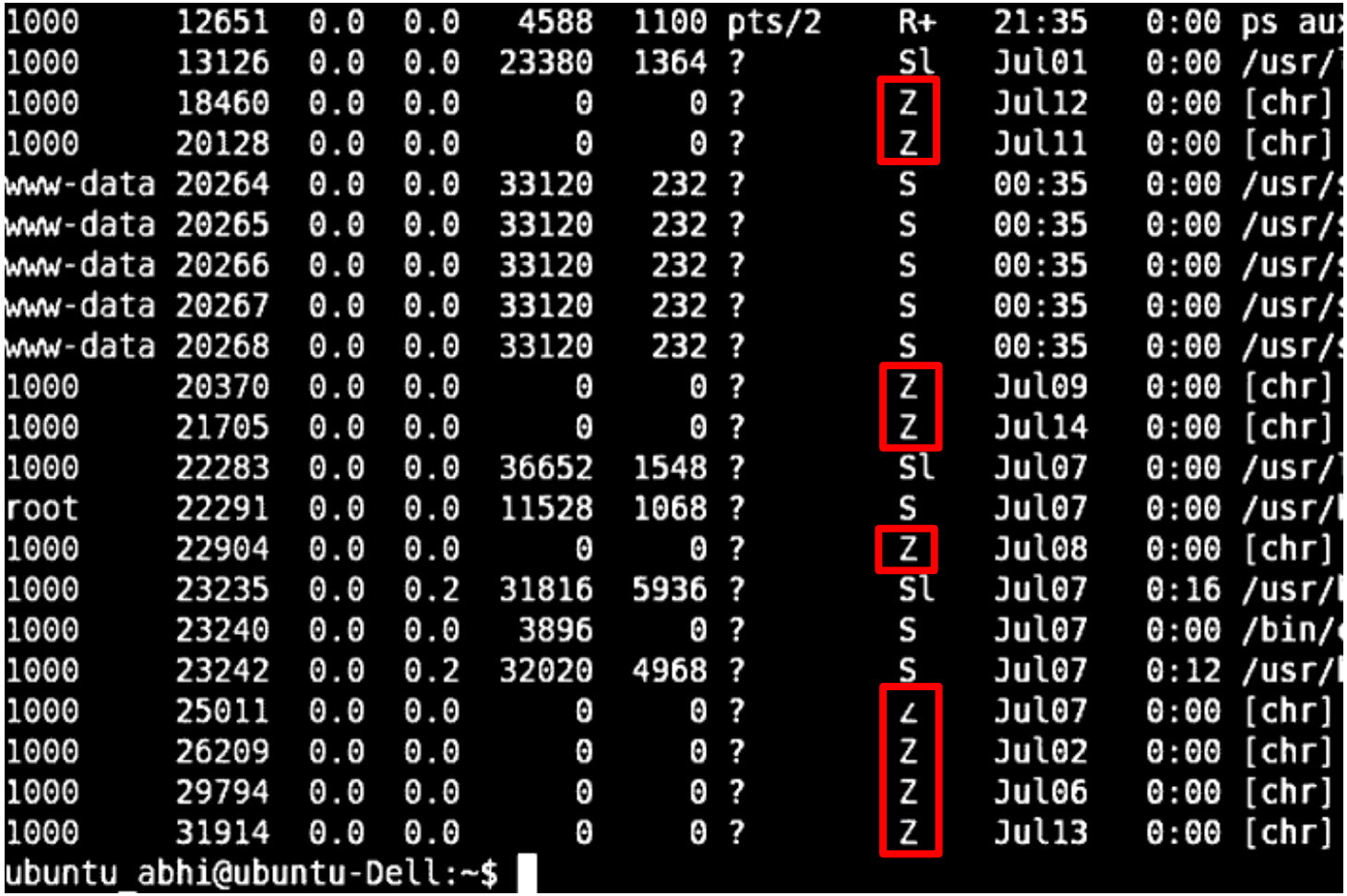
- PCB сохраняется пока родительский процесс не закрыл дескриптор дочернего процесса
  - CloseHandle(pHandle)
- Информацию о коде завершения можно получить системным вызовом
  - GetExitCodeProcess
- Дочерний процесс «не существует»
  - Нигде не отображается

PCB завершенного процесса Linux



- PCB сохраняется пока родительский процесс не запросил информацию о завершении дочернего процесса вызовами
  - wait / waitpid
    - Вызовы блокируют родительский процесс, пока дочерний не завершится
  - Завершившийся дочерний процесс **существует** в списке процессов
  - Дочерний процесс — ЗОМБИ!!!
  - Zombie — состояние процесса в *nix системах

Zombie


1000 12651 0.0 0.0 4588 1100 pts/2 R+ 21:35 0:00 ps au
1000 13126 0.0 0.0 23380 1364 ? Sl Jul01 0:00 /usr/
1000 18460 0.0 0.0 0 0 ? Z Jul12 0:00 [chr]
1000 20128 0.0 0.0 0 0 ? Z Jul11 0:00 [chr]
www-data 20264 0.0 0.0 33120 232 ? S 00:35 0:00 /usr/
www-data 20265 0.0 0.0 33120 232 ? S 00:35 0:00 /usr/
www-data 20266 0.0 0.0 33120 232 ? S 00:35 0:00 /usr/
www-data 20267 0.0 0.0 33120 232 ? S 00:35 0:00 /usr/
www-data 20268 0.0 0.0 33120 232 ? S 00:35 0:00 /usr/
1000 20370 0.0 0.0 0 0 ? Z Jul09 0:00 [chr]
1000 21705 0.0 0.0 0 0 ? Z Jul14 0:00 [chr]
1000 22283 0.0 0.0 36652 1548 ? Sl Jul07 0:00 /usr/
root 22291 0.0 0.0 11528 1068 ? S Jul07 0:00 /usr/
1000 22904 0.0 0.0 0 0 ? Z Jul08 0:00 [chr]
1000 23235 0.0 0.2 31816 5936 ? Sl Jul07 0:16 /usr/
1000 23240 0.0 0.0 3896 0 ? S Jul07 0:00 /bin/
1000 23242 0.0 0.2 32020 4968 ? S Jul07 0:12 /usr/
1000 25011 0.0 0.0 0 0 ? Z Jul07 0:00 [chr]
1000 26209 0.0 0.0 0 0 ? Z Jul02 0:00 [chr]
1000 29794 0.0 0.0 0 0 ? Z Jul06 0:00 [chr]
1000 31914 0.0 0.0 0 0 ? Z Jul13 0:00 [chr]
ubuntu_abhi@ubuntu-Dell:~$

PCB завершенного процесса Без родителя

- Если родительский процесс завершился, то ...

```
 ps
  PID  Uid      VmSize Stat Command
    1 root       1284 S    init
    2 root            S    [keventd]
    3 root            R    [ksoftirqd_CPU0]
    4 root            S    [kswapd]
    5 root            S    [bdflush]
    6 root            S    [kupdated]
    7 root            S    [mtdblockd]
   26 root            D    [adsl]
   32 root       1652 S    /usr/bin/cm_pc
   34 root       1204 S    /usr/sbin/thttpd -g -d /usr/www -u root -p 80 -c /cg
   35 root       1288 S    /usr/sbin/diap
   37 root       1284 S    init
   39 root       2848 S    /usr/bin/cm_logic -m /dev/ticfg -c /etc/config.xml
   41 root       1276 S    /usr/bin/cm_monitor
  265 root        632 S    /sbin/dproxy -c /etc/resolv.conf -d
  286 root        684 S    /usr/sbin/udhcpd /var/tmp/udhcpd.conf
 1139 root        616 S    /sbin/utelnetd
 1140 root       1288 S    -sh
 1153 root       1288 S    ash
 1182 root       2344 S    /usr/sbin/pppd plugin pppoe nas0 user username passw
 1196 root       1284 R    ps
```

своих дочерних процессов

Многоразовые операции

- Не изменяют список процессов

- Запуск процесса
- Приостановка процесса
- Блокирование процесса
- Разблокирование процесса

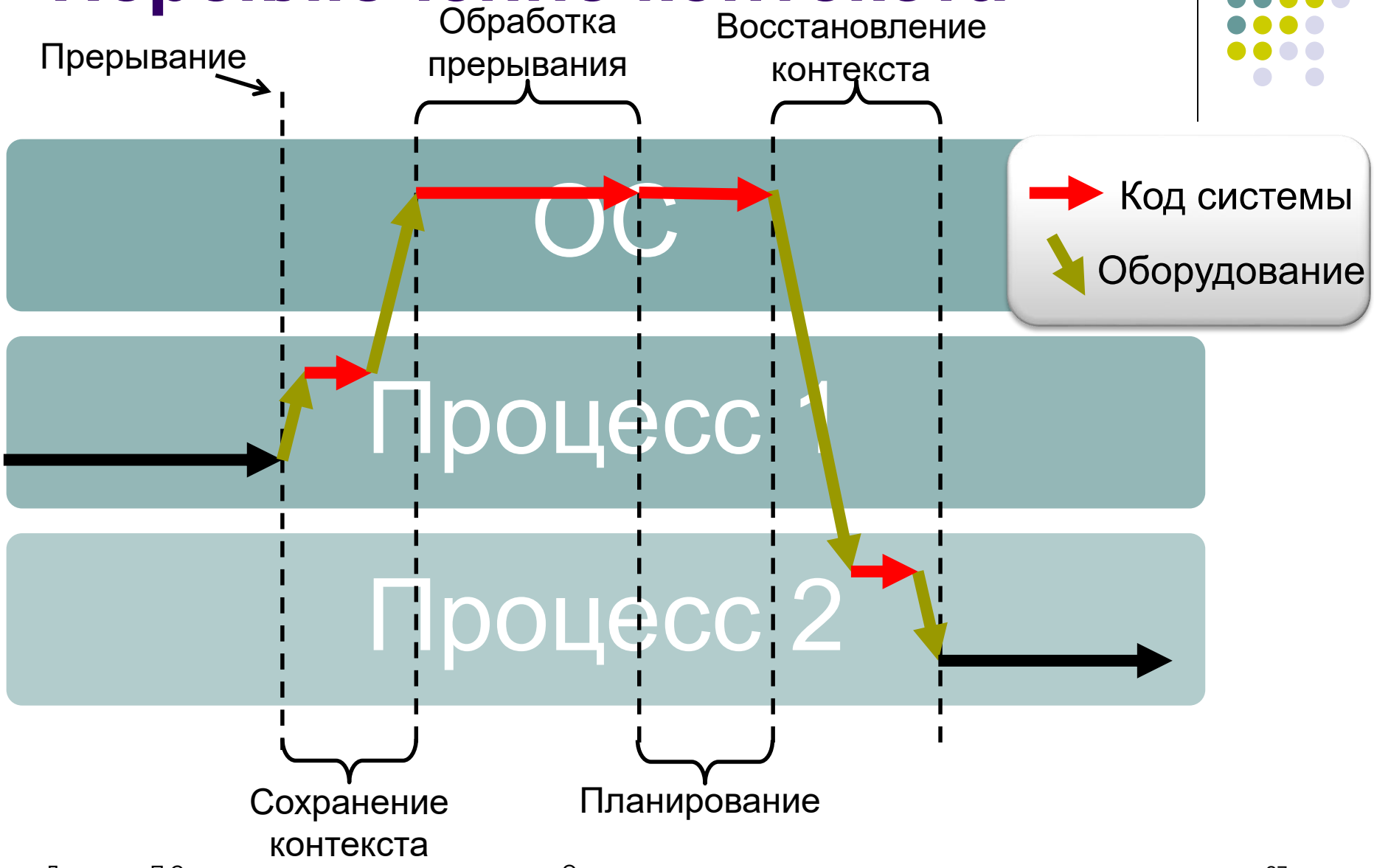
Запуск процесса

- Перевод из состояния «готовность» в «исполнение»
  - В ходе **планирования** процесс выбирается из очереди ожидания
  - Из PCB
    - Восстанавливается память
    - Восстанавливаются регистры
    - И другие составляющие контекста процесса

Многоразовые операции

- Приостановка процесса → «готовность»
  - В результате прерывания
  - Сохраняется регистровый и системный контекст
- Блокирование процесса → «ожидание»
  - Системный вызов для работы с ресурсом
  - ОС выполняет запрос, сохранив контекст
- Разблокирование процесса → «готовность»
  - При наступлении события
  - ОС определяет какие процессы ожидали данное событие

Переключение контекста



Вопросы?