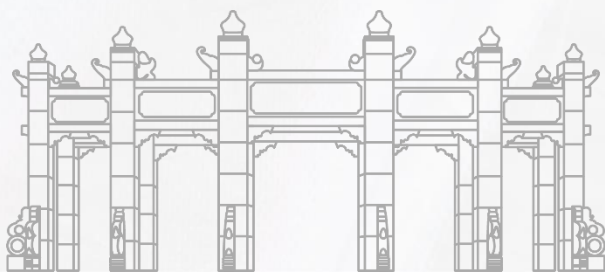
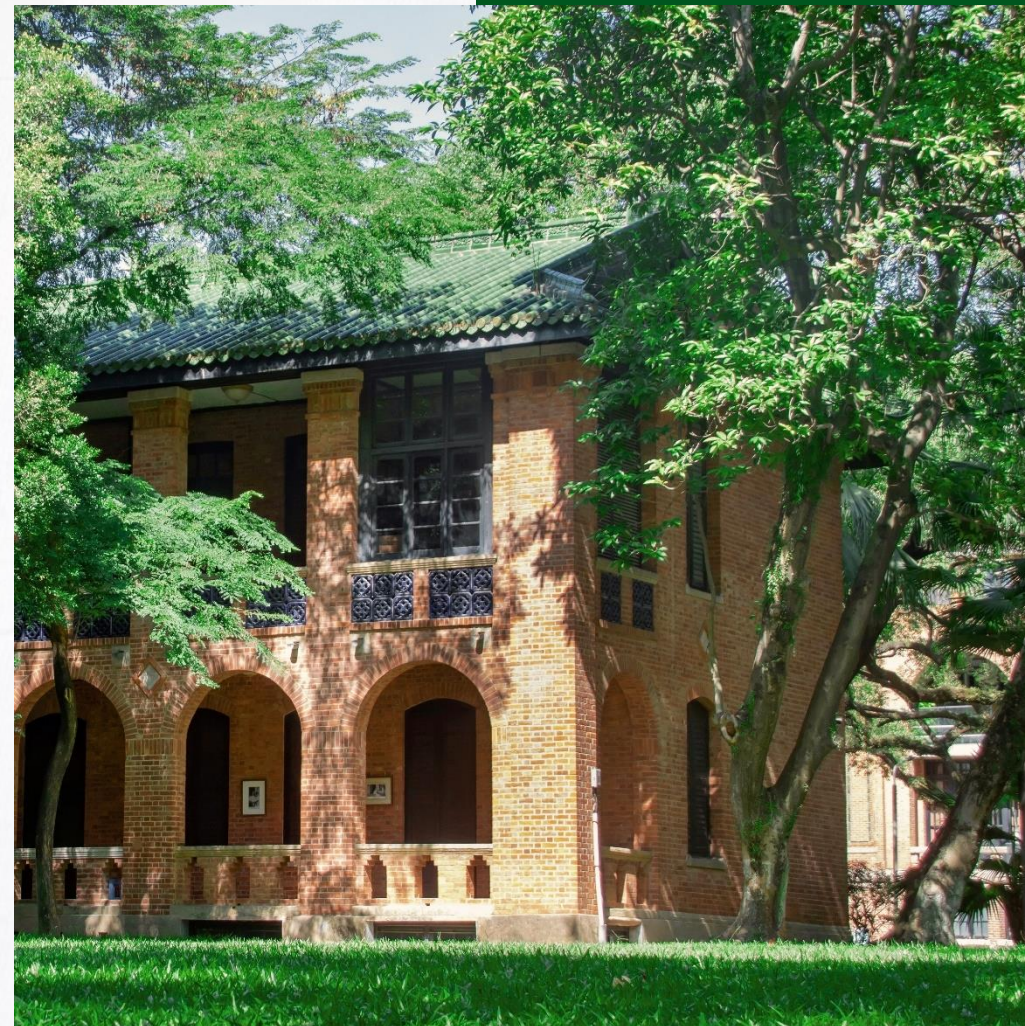


鸿蒙操作系统 进程与线程模型



● HarmonyOS的进程管理

1. 进程是LiteOS-a当中才有的概念， LiteOS-m只有task（线程）的概念。
2. OpenHarmony内核的进程模块可以给用户提供多个进程，实现了进程之间的切换和通信，帮助用户管理业务程序流程。
3. OpenHarmony内核的进程一共有22个优先级(10-31)，最高优先级为10，最低优先级为31。
4. 高优先级的进程可抢占低优先级进程，低优先级进程必须在**高优先级进程阻塞或结束后**才能得到调度。
5. 每一个用户态进程均拥有自己**独立**的进程空间，相互之间不可见，实现进程间隔离。
6. 用户态根进程init由内核态创建，其它用户态子进程均由init进程fork而来。

● HarmonyOS的进程模型

LiteOS-a中的进程控制块 (PCB)

- 进程与进程的关系，包括父进程、子进程、兄弟进程等。
- 进程与包含的线程的关系。
- 进程的调度控制信息，如优先级、剩余时间片等。
- 进程的资源参数，如内存等。
- 进程的权限信息，如管理员。

● HarmonyOS的进程模型

OpenHarmony中的**五种进程状态**

- 初始化 (Init)：该进程正在被创建。
- 就绪 (Ready)：该进程在就绪列表中，等待CPU调度。
- 运行 (Running)：该进程正在运行。
- 阻塞 (Pending)：该进程被阻塞挂起。本进程内所有的线程均被阻塞时，进程被阻塞挂起。
- 僵尸态 (Zombies)：该进程运行结束，等待父进程回收其控制块资源。

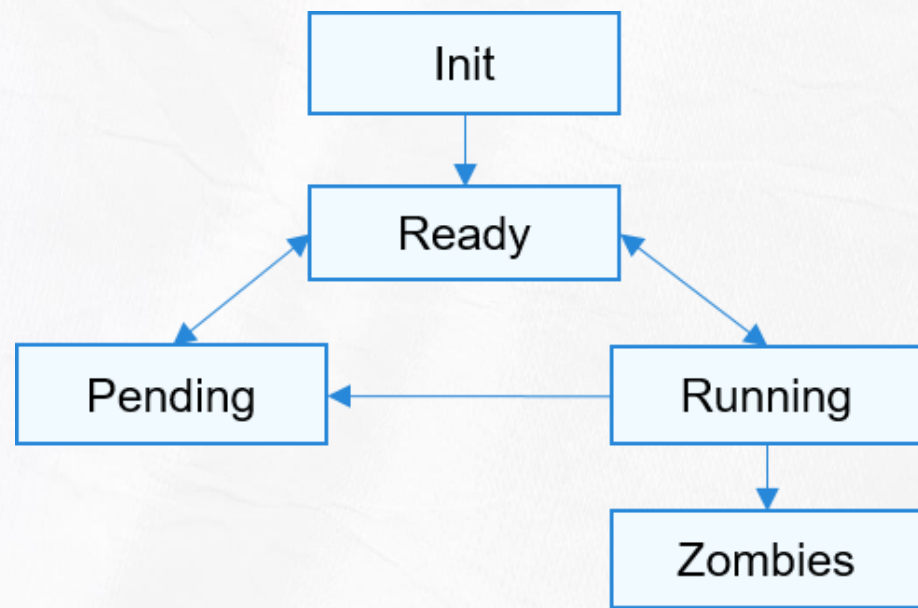


图3-1 进程状态迁移示意图

● HarmonyOS的进程状态转换

1. Init→Ready

- 进程创建或fork时，拿到该进程控制块后进入Init状态，处于进程初始化阶段，当进程初始化完成将进程插入调度队列，此时进程进入就绪状态。

2. Ready→Running

- 进程创建后进入就绪态，发生进程切换时，就绪列表中最高优先级的进程被执行，从而进入运行态。
- 若此时该进程中已处于就绪态，则进程从就绪列表删除，只处于运行态；若此时该进程中还有其它线程处于就绪态，则该进程依旧在就绪队列，此时进程的就绪态和运行态共存，但对外呈现的进程状态为运行态。

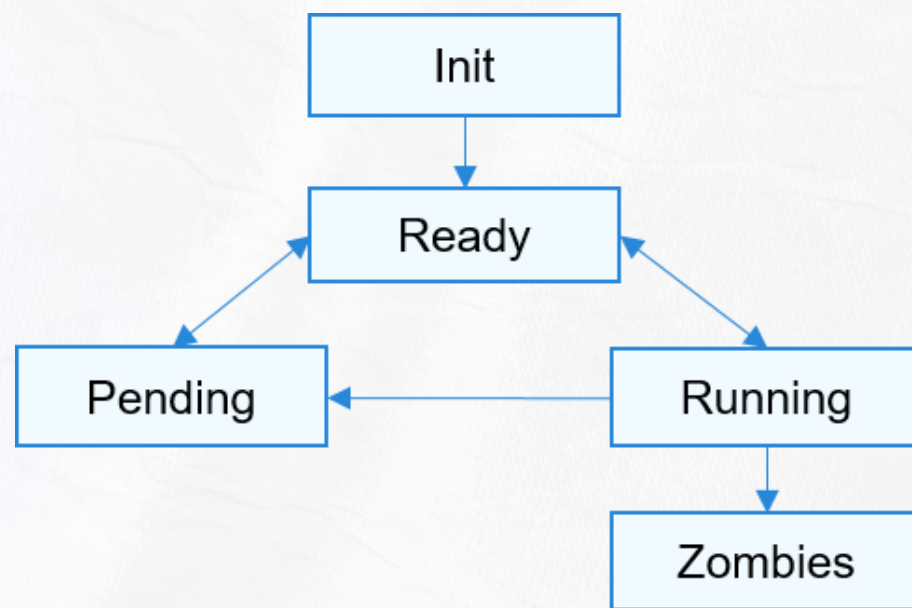


图3-1 进程状态迁移示意图

● HarmonyOS的进程状态转换

3. Running→Pending

- 进程在**最后一个线程**转为阻塞态时，进程内所有的线程**均处于阻塞态**，此时进程同步进入阻塞态，然后发生进程切换。

4. Pending→Ready

- 阻塞进程内的**任意线程**恢复就绪态时，进程被加入到就绪队列，同步转为就绪态。

5. Ready→Pending

- 进程内的**最后一个**就绪态线程转为阻塞态时，进程从就绪列表中删除，进程由就绪态转为阻塞态。

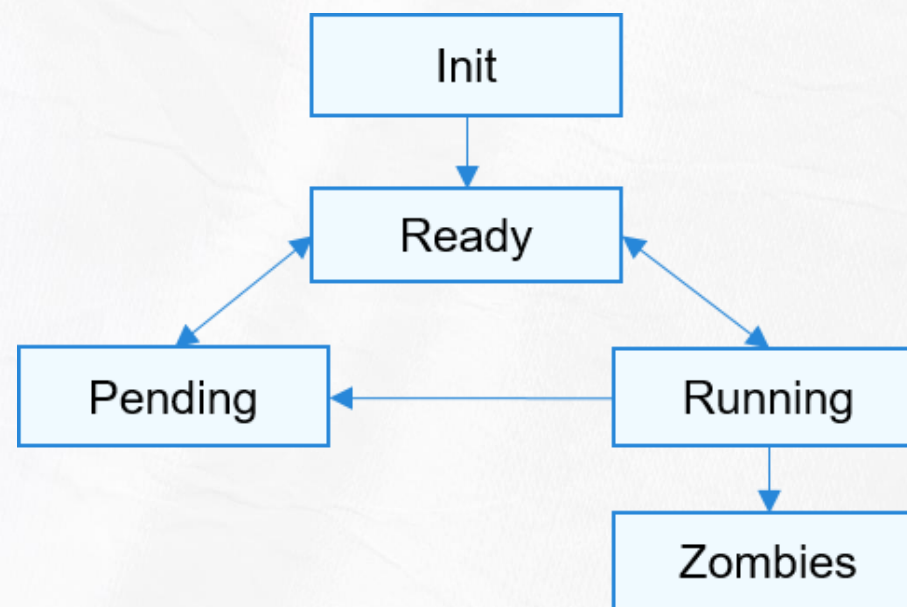


图3-1 进程状态迁移示意图

● HarmonyOS的进程状态转换

6. Running→Ready

- 有**更高优先级**的进程创建或者恢复后，会发生进程调度，此刻就绪列表中最高优先级进程变为运行态，那么原先运行的进程由运行态变为就绪态。
- 若进程的调度策略为 LOS_SCHED_RR（时间片轮转），且存在同一优先级的另一个进程处于就绪态，则该进程的**时间片消耗光**之后，该进程由运行态转为就绪态，另一个同优先级的进程由就绪态转为运行态。

7. Running→Zombies

- 当进程的主线程或所有线程运行结束后，进程由运行态转为僵尸态，等待父进程回收资源。

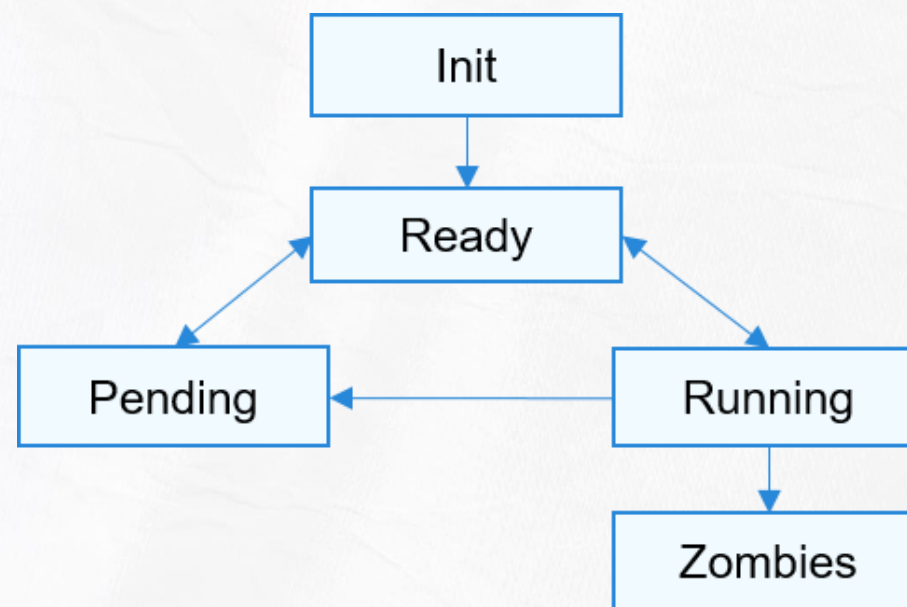


图3-1 进程状态迁移示意图

● HarmonyOS的进程运行机制

1. OpenHarmony提供的进程模块主要用于实现用户态进程的隔离。
 - 支持**用户态进程**的创建、退出、资源回收、设置/获取调度参数、获取进程ID、设置/获取进程组ID等功能。
2. 用户态进程通过fork父进程而来，fork进程时会将**父进程的进程虚拟内存空间clone到子进程**，子进程实际运行时通过**写时复制机制**将父进程的内容**按需复制**到子进程的虚拟内存空间。
3. 进程只是**资源管理单元**，实际运行是由进程内的各个线程完成的，不同进程内的线程相互切换时**会进行进程空间的切换**。

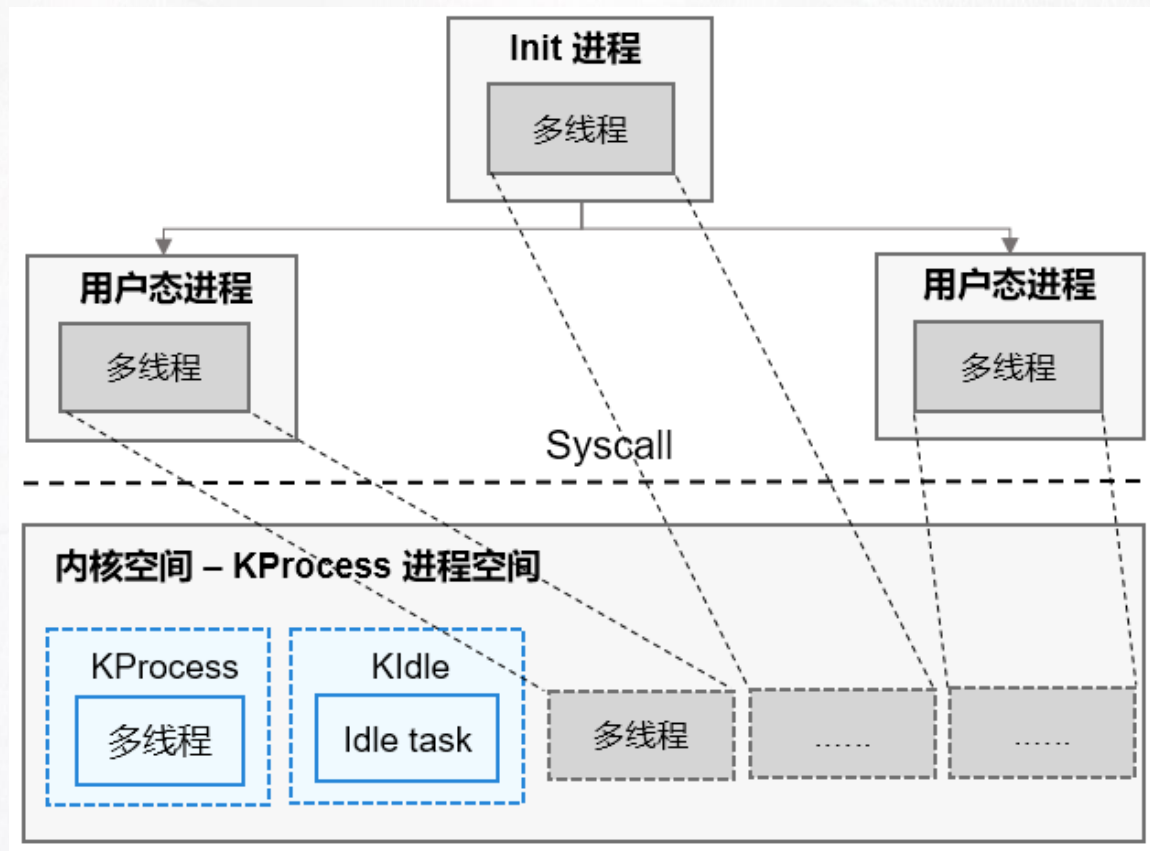


图3-2 进程管理示意图

● HarmonyOS的线程管理

1. 线程既存在于LiteOS-a，也存在于LiteOS-m。
2. 系统的角度看，线程是竞争系统资源的**最小运行单元**。线程可以使用或等待CPU、使用内存空间等系统资源，并独立于其它线程运行。
3. OpenHarmony内核每个进程内的线程**独立运行、独立调度**，**当前进程内线程的调度不受其它进程内线程的影响**。
4. OpenHarmony内核中的线程采用**抢占式调度**机制，同时支持时间片轮转调度和FIFO调度方式。
5. OpenHarmony 内核的任务一共有32个优先级(0-31)，最高优先级为0，最低优先级为31。
6. 当前进程内高优先级的线程可抢占当前进程内低优先级线程，当前进程内低优先级线程必须在当前进程内高优先级线程**阻塞或结束**后才能得到调度。

● HarmonyOS的线程模型

1. LiteOS-m的线程控制块 (TCB)

- 运行控制参数，如堆栈、优先级、状态、函数入口等。
- IPC进程间通信的相关参数，如持有的信号量等。

2. LiteOS-a的线程控制块 (TCB)

- 在LiteOS-m的线程控制块的基础上增加了如下参数：
- 支持复杂调度模式的参数
- 进程相关的数据，如归属进程号。
- 多核SMP的支持参数。
- 多进程相关的参数。

● HarmonyOS的线程模型

OpenHarmony中的**五种线程状态**

- 初始化 (Init) : 该线程正在被创建。
- 就绪 (Ready) : 该线程在就绪列表中, 等待CPU调度。
- 运行 (Running) : 该线程正在运行。
- 阻塞 (Blocked) : 该线程被阻塞挂起。Blocked状态包括: pend(因为锁、事件、信号量等阻塞)、suspend (主动 pend) 、delay(延时阻塞)、pendtime(因为锁、事件、信号量时间等超时等待)。
- 退出 (Exit) : 该线程运行结束, 等待父线程回收其控制块资源。

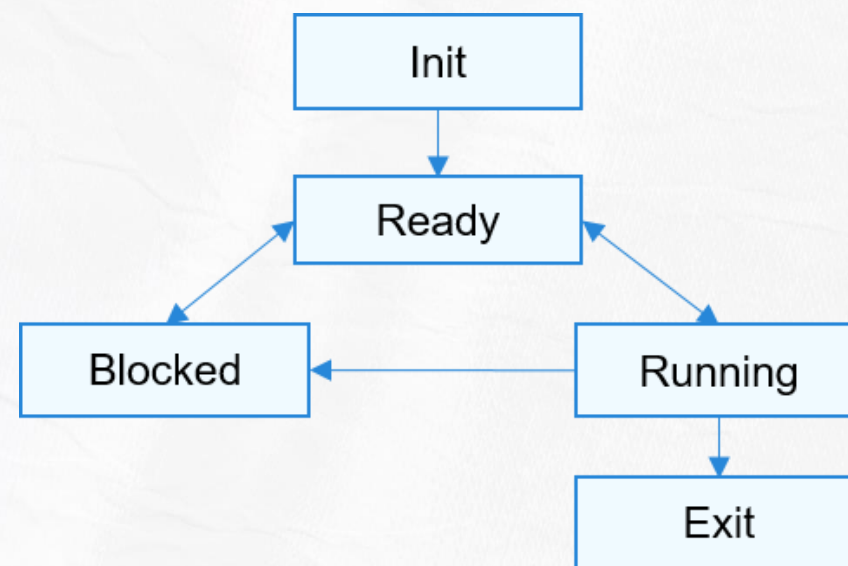


图3-3 状态迁移示意图

● HarmonyOS的线程状态转换

1. Init→Ready

线程创建拿到控制块后为Init状态，处于线程初始化阶段，当线程初始化完成将线程插入调度队列，此时线程进入就绪状态。

2. Ready→Running

任务创建后进入就绪态，发生任务切换时，就绪列表中最高优先级的任务被执行，从而进入运行态，此刻该任务从就绪列表中删除。

3. Running→Blocked

正在运行的任务发生阻塞（挂起、延时、读信号量等）时，任务状态由运行态变成阻塞态，然后发生任务切换，运行就绪列表中剩余最高优先级任务。

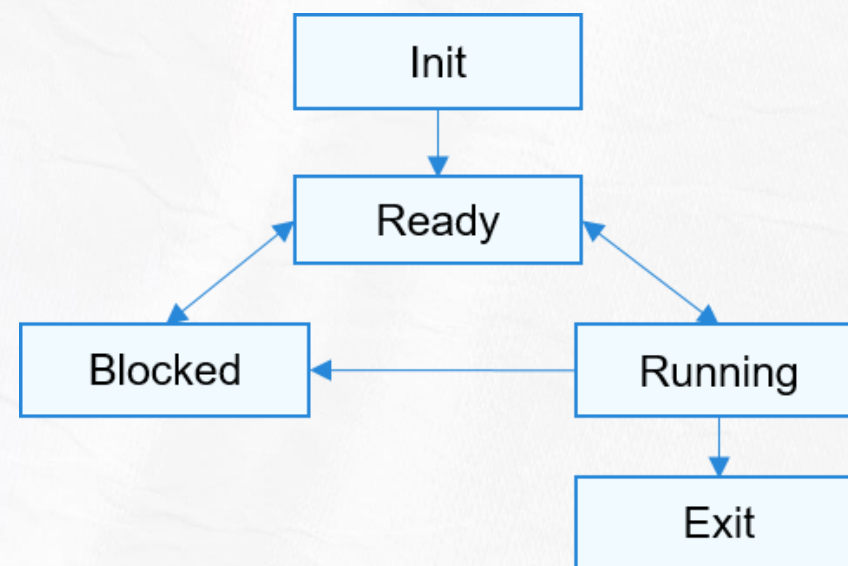


图3-3 状态迁移示意图

● HarmonyOS的线程状态转换

4. Blocked→Ready

阻塞的任务被恢复后（任务恢复、延时时间超时、读信号量超时或读到信号量等），此时被恢复的任务会被加入就绪列表，从而由阻塞态变成就绪态。

5. Ready→Blocked

任务也有可能~~在就绪态时被阻塞（挂起）~~，此时任务状态会由就绪态转变为阻塞态，该任务从就绪列表中删除，不会参与任务调度，直到该任务被恢复。

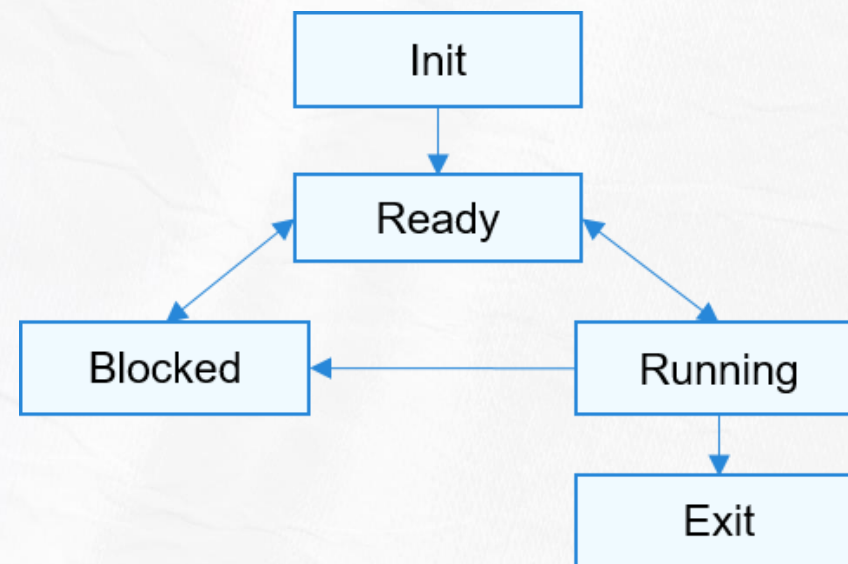


图3-3 状态迁移示意图

● HarmonyOS的线程状态转换

6. Running→Ready

有更高优先级任务创建或者恢复后，会发生任务调度，此刻就绪列表中**最高优先级任务变为运行态**，那么原先运行的任务由运行态变为就绪态，并加入就绪列表中。

7. Running→Exit

运行中的任务运行结束，任务状态由运行态变为退出态。若为设置了分离属性（由头文件 `los_task.h` 中的宏定义 `LOS_TASK_STATUS_DETACHED` 设置）的任务，运行结束后将直接销毁。

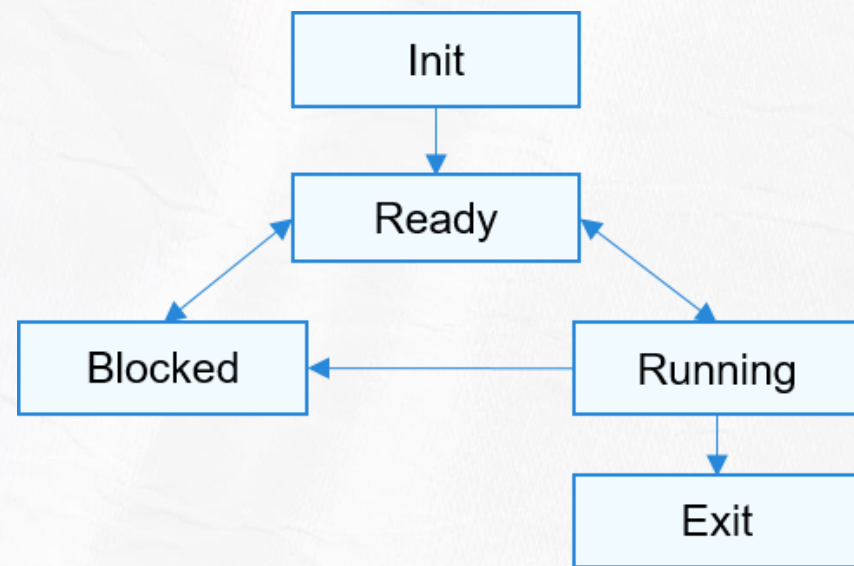


图3-3 状态迁移示意图

● 用户对进程和线程的操作

1. 对进程的操作

- 进程创建后，用户只能操作自己进程空间的资源，无法操作其它进程的资源（共享资源除外）。
- 用户态允许进程挂起，恢复，延时等操作，同时也可以设置用户态进程调度优先级和调度策略，获取进程调度优先级和调度策略。
- 进程结束的时候，进程会主动释放持有的进程资源，但持有的进程pid资源需要父进程通过wait/waitpid或父进程退出时回收。

2. 对线程的操作

- 线程创建后，用户态可以执行线程调度、挂起、恢复、延时等操作，同时也可以设置线程优先级和调度策略，获取线程优先级和调度策略。

● 用户对进程和线程的操作

1. 对进程操作的常用接口

- LOS_GetCurrProcessID: 获取当前进程的进程ID
- LOS_GetProcessGroupID: 获取指定进程的进程组ID
- LOS_GetUserID: 获取当前进程的用户ID
- LOS_GetGroupID: 获取当前进程的用户组ID
- LOS_GetProcessScheduler: 获取指定进程的调度策略
- LOS_SetProcessScheduler: 设置指定进程的调度参数, 包括优先级和调度策略
- LOS_GetUsedPIDList: 获得已使用的进程ID列表
- LOS_Fork: 创建子进程
- LOS_Wait: 等待子进程结束并回收子进程
- LOS_Exit: 退出进程

● 用户对进程和线程的操作

1. 对线程 (Task) 操作的常用接口

- LOS_TaskCreate: 创建任务
- LOS_TaskCreateOnly: 创建任务并阻塞, 任务恢复前不会将其加入就绪队列中
- LOS_TaskDelete: 删除指定的任务, 回收其任务控制块和任务栈所消耗的资源
- LOS_TaskResume: 恢复挂起的任务
- LOS_TaskJoin: 阻塞当前任务, 等待指定任务运行结束并回收其资源
- LOS_TaskLock: 锁定任务调度, 阻止任务切换
- LOS_TaskUnlock: 解锁任务调度。通过该接口可以使任务锁数量减1, 若任务多次加锁, 那么任务调度在锁数量减为0时才会完全解锁
- LOS_TaskPriSet: 设置指定任务的优先级
- LOS_CurTaskPriSet: 设置当前正在运行的任务的优先级



中山大學
SUN YAT-SEN UNIVERSITY

谢谢观看

SUN YAT-SEN UNIVERSITY