■コールバック（callback）

●通知と代行

自分のやることを忘れないために以下のような方法があります。

(1) 実行を忘れないように実行時期を誰かに教えてもらう
(2) 実行そのものを誰かに代行してもらう

ここで、クリーニングの受け取りについて考えて見ます。
Aさんはクリーニング屋さんにクリーニング完了のタイミングでメールや電話で
連絡してもらえるサービスを利用しています。
クリーニング屋さんからの連絡後、自分の空き時間でクリーニング屋さんに
取りに行くことになります。

Bさんはクリーニング屋さんと提携している代行業者にクリーニングの
受け取りを依頼しています。
業者にはクリーニングされた衣類をクリーニング屋さんからの受け取り後に
自宅の宅配ボックスや勤務先に届けてもらいます。
届け先や完了時の連絡方法などはオプションでその都度変更が可能です。

Aさんの場合は連絡してもらうだけで、実行は常に自分の責任で行います。
面倒ですが自分で状態やスケジュールを調整して受け取りを実行するので、
行き違いは発生しません。

Bさんの場合、自分の都合で自ら実行するような場合や複数の業者を使い分ける
ような場合には自分と業者や業者同士の調整をするのに却って手間がかかって
しまいます。

●イベント通知の手法と開発効率

コールバック(callback)はコールバック関数(callback function)の省略形
として使われる場合やコールバックルーチン(callback routine)の省略形で
使われる場合があります。
本来は電話をかけた相手から依頼した結果を折り返し連絡をもらうような
場合に使われる用語です。
開発実装上は以下のような使われ方をします。

(a) 依頼先のイベント通知処理に対して関数ポインタを引数として渡すことにより
　　自分で作った関数を依頼先に登録できる
(b) イベント通知の種類毎に自分で作った関数を使い分けることができる
(c) イベントの種類を一つのコールバック関数の引数でもらえる仕組みもある
(d) 非同期処理の場合にのみ使われる

多くの場合、コールバックの仕組みはOSが提供する場合やミドルウェアの
特定のSDK（Software Development Kit）としての
API(Application Program Interface)が提供する場合があります。
自分が作りたいプログラム（アプリケーション）と汎用的なSDKが提供する機能を
分離することが主な目的となります。
このような機能の分離はソフトウェアの可搬性や開発効率を高めます。

●組込みとPC用OSとの違い

WindowsなどのPC用のOS上で動作するアプリケーションを開発する場合、
明確な開発に関する作法がSDKやOSにより決まっています。
自分が作っているアプリケーションは特に意識しない限り、
言語処理系とSDKの連携で比較的簡単に矛盾のないアプリケーション開発が
可能になっています。
中身の仕組みは別としても自分（アプリケーション）の責任範囲が
明確な開発が出来るようになっています。
実行コンテキストについての配慮や資源同期の配慮も不要です。
アプリケーション開発者視点で考えるとAさんのパターンと同様です。

ところが組込みシステムで採用するようなコールバックの仕組みはまちまちです。
「コールバック」という用語が一人歩きするだけで実装の仕組みは決まっていません。
開発する会社やチームの違いによっても実装の違いがあり得ます。
これはBさんと業者の関係に似ています。

以下の留意が必要です。

・非同期イベント検出の起点は割り込み処理
・繰り返し状態を見るようなポーリング処理でイベントを検出する場合もある
・イベント検出後のコールバック関数の実行は割り込み処理の場合や
　自分以外の他のタスクである場合が多い
・SDKとして提供されてソースコードが見えない場合はコールバック関数の
　実行コンテキストについて把握しておく必要がある
・アプリケーションとコールバックの実行コンテキストが異なる場合は
　資源の排他制御が必要

このように、アプリケーション開発者が一定の基準でコールバックを
捉えていたとしても実行コンテキストや資源同期の仕方が同じであるということが
保証できない場合があります。
これはOSを使わない組込みシステム、μITRONを使った組込みシステムでの
開発特有の問題です。