第6章 デッドロック 演習問題

1. 政治から取られたデッドロックの例を挙げなさい。
2. コンピュータ研究室の個々のPCで作業している学生が、ハードディスク上のスプール領域にファイルをスプールするサーバーに印刷ファイルを送信します。プリントスプールのディスク領域が限られている場合、どのような状況でデッドロックが発生する可能性がありますか。そのデッドロックはどのように回避できますか。
3. 前の問題において、どのリソースがプリエンプティブル（横取り可能）で、どれがノンプリエンプティブル（横取り不可能）ですか。
4. 図6-1では、リソースは取得した順序とは逆の順序で返却されています。これを別の順序で返却しても同様にうまく機能しますか。
5. 4つの条件（相互排他、保持と待機、横取りなし、循環待ち）は、リソースデッドロックが発生するために必要です。これらの条件がリソースデッドロックが発生するための十分条件ではないことを示す例を挙げなさい。これらの条件がリソースデッドロックの発生に十分となるのはどのような場合ですか。
6. 都市の街路は、グリッドロックと呼ばれる循環的な閉塞状態に陥りやすいです。これは、交差点が車で塞がれ、その後ろの車が塞がれ、さらにその前の交差点に入ろうとする車が塞がれるといった状況です。都市の一区画を囲むすべての交差点が、循環的に対向交通を塞ぐ車両で満たされます。グリッドロックはリソースデッドロックであり、競合同期の問題です。ニューヨーク市の予防アルゴリズムは「交差点を塞ぐな（don’t block the box）」と呼ばれ、交差点の先のスペースが利用可能でない限り、車が交差点に入ることを禁止します。これはどの予防アルゴリズムに該当しますか。グリッドロックに対する他の予防アルゴリズムを提案できますか。
7. 4台の車がそれぞれ4つの異なる方向から同時に交差点に近づくとします。交差点の各角には一時停止標識があります。交通規則では、2台の車が隣接する一時停止標識に同時に近づいた場合、左側の車が右側の車に道を譲らなければならないと仮定します。したがって、4台の車がそれぞれの一時停止標識に近づくと、各車は左側の車が進むのを（無期限に）待ちます。この異常は通信デッドロックですか。それともリソースデッドロックですか。
8. あるリソースタイプの複数のユニットと、別のタイプの一つのユニットが関与するリソースデッドロックは可能ですか。もしそうなら、例を挙げなさい。
9. 図6-3はリソースグラフの概念を示しています。不正なグラフ、つまり、我々が使用してきたリソース利用のモデルを構造的に違反するグラフは存在しますか。もしそうなら、その例を挙げなさい。
10. 図6-4を考えます。ステップ(o)でCがRではなくSを要求したとします。これはデッドロックにつながりますか。SとRの両方を要求した場合はどうなりますか。
11. システムにリソースデッドロックが存在するとします。デッドロック状態にあるプロセスの集合に、対応するリソース割り当てグラフの循環鎖に含まれていないプロセスが含まれうることを示す例を挙げなさい。
12. トラフィックを制御するために、ネットワークルータAは定期的に隣接するルータBにメッセージを送り、処理可能なパケット数を増減させるよう指示します。ある時点で、ルータAはトラフィックで溢れ、Bにトラフィックの送信を停止するようメッセージを送ります。これは、Bが送信できるバイト数（Aのウィンドウサイズ）を0に指定することで行います。トラフィックの急増が減少すると、Aは新しいメッセージを送り、Bに送信を再開するよう指示します。これはウィンドウサイズを0から正の数に増やすことで行います。そのメッセージが失われました。記述されたように、どちらの側も送信しなくなります。これはどのタイプのデッドロックですか。
13. ダチョウアルゴリズムの議論では、プロセステーブルのスロットや他のシステムテーブルが満杯になる可能性について言及されています。システム管理者がそのような状況から回復できるようにする方法を提案できますか。
14. 4つのプロセスP1, P2, P3, P4と5種類のリソースRS1, RS2, RS3, RS4, RS5を持つシステムの以下の状態を考えます： E = (2 4 1 4 4) A = (0 1 0 2 1) C = [,,,] R = [,,,] セクション6.4.2で説明されたデッドロック検出アルゴリズムを使用して、システムにデッドロックがあることを示しなさい。デッドロック状態にあるプロセスを特定しなさい。
15. 前の問題のデッドロックからシステムがどのように回復できるかを、以下の方法で説明しなさい。
    a. 横取りによる回復
    b. ロールバックによる回復

c. プロセス強制終了による回復

16. 図6-6において、あるiに対して`Cij + Rij > Ej`であるとします。これはシステムにどのような影響を及ぼしますか。
17. 図6-8の軌跡はすべて水平または垂直です。斜めの軌跡も可能となるような状況を想像できますか。
18. 図6-8のリソース軌跡の仕組みは、3つのプロセスと3つのリソースを持つデッドロック問題を示すためにも使用できますか。もしそうなら、どのようにできますか。もしできなければ、なぜですか。
19. 理論的には、リソース軌跡グラフはデッドロックを回避するために使用できます。巧妙なスケジューリングによって、オペレーティングシステムは危険な領域を避けることができます。これを実際に実行するための実用的な方法はありますか。
20. システムがデッドロック状態でもなく安全状態でもない状態になることは可能ですか。もしそうなら、例を挙げなさい。もしできなければ、すべての状態がデッドロック状態か安全状態のどちらかであることを証明しなさい。
21. 図6-11(b)を注意深く見てください。もしDがもう1ユニット要求した場合、これは安全状態につながりますか、それとも危険状態につながりますか。要求がDではなくCから来た場合はどうなりますか。
22. あるシステムには2つのプロセスと3つの同一のリソースがあります。各プロセスは最大2つのリソースを必要とします。デッドロックは可能ですか。あなたの答えを説明しなさい。
23. 前の問題を再び考えますが、今度はp個のプロセスがそれぞれ最大m個のリソースを必要とし、合計r個のリソースが利用可能であるとします。システムをデッドロックフリーにするためには、どのような条件が満たされなければなりませんか。
24. 図6-12において、プロセスAが最後のテープドライブを要求したとします。この行動はデッドロックにつながりますか。
25. m個のリソースクラスとn個のプロセスを持つシステムで銀行家のアルゴリズムが実行されています。mとnが大きい極限において、ある状態が安全かどうかをチェックするために実行しなければならない操作の数は、`m^a * n^b`に比例します。aとbの値は何ですか。
26. あるシステムには4つのプロセスと5つの割り当て可能なリソースがあります。現在の割り当てと最大の必要量は以下の通りです： プロセス | 割り当て | 最大 | 利用可能 —|—|—|— A | 1 0 2 1 1 | 1 1 2 1 3 | 0 0 x 1 1 B | 2 0 1 1 0 | 2 2 2 1 0 | C | 1 1 0 1 0 | 2 1 3 1 0 | D | 1 1 1 1 0 | 1 1 2 2 1 | これが安全状態であるためのxの最小値は何ですか。
27. 循環待ちをなくす一つの方法は、プロセスが一度に一つのリソースしか保有できないという規則を設けることです。多くの場合、この制約が受け入れられないことを示す例を挙げなさい。
28. 2つのプロセスAとBが、それぞれデータベース内の3つのレコード1, 2, 3を必要とします。もしAが1, 2, 3の順で要求し、Bも同じ順で要求すれば、デッドロックは起こりえません。しかし、もしBが3, 2, 1の順で要求すれば、デッドロックは可能です。3つのリソースがある場合、各プロセスがそれらを要求できる組み合わせは3!、つまり6通りあります。すべての組み合わせのうち、デッドロックフリーであることが保証される割合はどれくらいですか。
29. メールボックスを使用する分散システムには、sendとreceiveという2つのIPCプリミティブがあります。後者のプリミティブは受信するプロセスを指定し、そのプロセスからのメッセージが利用可能でなければ、他のプロセスからのメッセージが待機していてもブロックします。共有リソースはありませんが、プロセスは他の事柄について頻繁に通信する必要があります。デッドロックは可能ですか。議論しなさい。
30. 電子資金移動システムには、次のように動作する何百もの同一のプロセスがあります。各プロセスは、金額、入金先口座、出金元口座を指定する入力行を読み取ります。その後、両方の口座をロックして送金し、完了したらロックを解除します。多くのプロセスが並行して実行されるため、口座xをロックしたプロセスが、yをロックしようとしてできなくなる（yが今xを待っているプロセスによってロックされているため）というデッドロックの危険性が非常に高いです。デッドロックを回避する仕組みを考案しなさい。トランザクションが完了するまで口座レコードを解放してはなりません。（言い換えれば、一方の口座をロックし、他方がロックされている場合はすぐにそれを解放するという解決策は許可されません。）
31. デッドロックを防ぐ一つの方法は、「保持と待機」の条件をなくすことです。本文では、新しいリソースを要求する前に、プロセスは既に保持しているリソースをすべて解放しなければならない（それが可能であると仮定して）と提案されました。しかし、そうすると、新しいリソースは得られても、既存のリソースの一部を競合するプロセスに失う危険が生じます。この仕組みに対する改善案を提案しなさい。
32. デッドロックに取り組むよう割り当てられたコンピュータ科学の学生が、デッドロックをなくすための次の素

晴らしい方法を考え出しました。プロセスがリソースを要求するとき、時間制限を指定します。リソースが利用できずにプロセスがブロックされた場合、タイマーが開始されます。時間制限を超えると、プロセスは解放され、再び実行が許可されます。もしあなたが教授なら、この提案にどのような評価を与え、その理由は何ですか。

33. 主メモリユニットは、スワッピングや仮想メモリシステムで横取り（プリエンプション）されます。プロセッサは、タイムシェアリング環境で横取りされます。これらの横取り方法は、リソースデッドロックを処理するために開発されたと思いますか、それとも他の目的のためですか。それらのオーバーヘッドはどの程度高いですか。
34. デッドロック、ライブロック、飢餓状態（スターベーション）の違いを説明しなさい。
35. 2つのプロセスが、ディスクにアクセスするためのメカニズムを再配置し、読み取りコマンドを有効にするためのシークコマンドを発行していると仮定します。各プロセスは読み取りを実行する前に中断され、他方がディスクアームを動かしたことに気づきます。各プロセスはその後シークコマンドを再発行しますが、再び他方によって中断されます。このシーケンスが絶えず繰り返されます。これはリソースデッドロックですか、それともライブロックですか。この異常を処理するためにどのような方法を推奨しますか。
36. ローカルエリアネットワークは、CSMA/CDと呼ばれるメディアアクセス方式を利用します。これは、バスを共有するステーションがメディアを感知し、送信だけでなく衝突も検出できるものです。イーサネットプロトコルでは、共有チャネルを要求するステーションは、メディアがビジーであると感知した場合はフレームを送信しません。そのような送信が終了すると、待機していたステーションがそれぞれフレームを送信します。同時に送信された2つのフレームは衝突します。もしステーションが衝突検出後すぐに繰り返し再送信すると、それらは無期限に衝突し続けます。
    a. これはリソースデッドロックですか、それともライブロックですか。
    b. この異常に対する解決策を提案できますか。
    c. このシナリオで飢餓状態は発生しますか。
37. あるプログラムには、協調と競合のメカニズムの順序に誤りがあり、その結果、コンシューマプロセスが空のバッファでブロックする前にミューテックス（相互排他セマフォ）をロックしてしまいます。プロデューサプロセスは、空のバッファに値を置いてコンシューマを起こす前に、そのミューテックスでブロックします。したがって、両方のプロセスは永遠にブロックされ、プロデューサはミューテックスのアンロックを待ち、コンシューマはプロデューサからのシグナルを待ちます。これはリソースデッドロックですか、それとも通信デッドロックですか。その制御方法を提案しなさい。
38. シンデレラと王子様が離婚することになりました。財産を分けるために、彼らは次のアルゴリズムに合意しました。毎朝、それぞれが相手の弁護士に手紙を送り、一つの財産項目を要求できます。手紙が届くのに1日かかるため、もし両者が同じ日に同じ項目を要求したことがわかった場合、翌日には要求をキャンセルする手紙を送ることに合意しました。彼らの財産の中には、犬のウーファー、ウーファーの犬小屋、カナリアのツィーター、ツィーターの鳥かごがあります。動物たちは自分の家を愛しているので、動物とその家を分離するような財産分割は無効であり、分割全体を最初からやり直す必要があると合意されています。シンデレラも王子様もウーファーを必死に欲しがっています。彼らが（別々の）休暇に出かけられるように、各配偶者は交渉を処理するためのパーソナルコンピュータをプログラムしました。休暇から戻ってくると、コンピュータはまだ交渉中です。なぜですか。デッドロックは可能ですか。飢餓状態は可能ですか。あなたの答えを議論しなさい。
39. 人類学を専攻し、コンピュータ科学を副専攻とする学生が、アフリカのヒヒにデッドロックについて教えることができるかどうかを調べる研究プロジェクトに着手しました。彼は深い峡谷を見つけ、その上にロープを渡し、ヒヒが手で渡れるようにしました。複数のヒヒが同時に渡ることができますが、全員が同じ方向に進んでいる場合に限ります。もし東向きと西向きのヒヒが同時にロープに乗ると、デッドロックが発生します（ヒヒは真ん中で動けなくなります）。なぜなら、峡谷の上にぶら下がっている状態で、一方のヒヒが他方のヒヒを乗り越えることは不可能だからです。ヒヒが峡谷を渡りたい場合、反対方向に渡っているヒヒがいないことを確認しなければなりません。セマフォを使用してデッドロックを回避するプログラムを書きなさい。一連の東向きのヒヒが西向きのヒヒを無期限に待たせることについては心配しなくてよいです。
40. 前の問題を繰り返しますが、今度は飢餓状態を回避してください。東に渡りたいヒヒがロープに到着し、西に渡っているヒヒを見つけた場合、彼はロープが空になるまで待ちます。しかし、少なくとも一匹のヒヒが反対方向に渡るまで、それ以上西向きのヒヒが渡り始めることは許されません。
41. 銀行家のアルゴリズムのシミュレーションをプログラムしなさい。あなたのプログラムは、各銀行のクライア

ントを順に巡回し、要求を尋ね、それが安全か危険かを評価する必要があります。要求と決定のログをファイルに出力しなさい。

42. 各タイプのリソースが複数ある場合のデッドロック検出アルゴリズムを実装するプログラムを書きなさい。あなたのプログラムは、ファイルから以下の入力を読み取る必要があります：プロセスの数、リソースタイプの数、存在する各リソースタイプの数（ベクトルE）、現在の割り当て行列C（最初の行、次に2番目の行、…）、要求行列R（最初の行、次に2番目の行、…）。プログラムの出力は、システムにデッドロックがあるかどうかを示す必要があります。デッドロックがある場合、プログラムはデッドロック状態にあるすべてのプロセスのIDをプリントアウトする必要があります。
43. リソース割り当てグラフを使用してシステムにデッドロックがあるかどうかを検出するプログラムを書きなさい。あなたのプログラムは、ファイルから以下の入力を読み取る必要があります：プロセスの数とリソースの数。各プロセスについて、4つの数値を読み取ります：現在保持しているリソースの数、保持しているリソースのID、現在要求しているリソースの数、要求しているリソースのID。プログラムの出力は、システムにデッドロックがあるかどうかを示す必要があります。デッドロックがある場合、プログラムはデッドロック状態にあるすべてのプロセスのIDをプリントアウトする必要があります。
44. ある国では、2人が会うとお互いにお辞儀をします。プロトコルは、一方が先にお辞儀をし、もう一方がお辞儀をするまでそのままでいるというものです。もし同時にお辞儀をすると、両者とも永遠にお辞儀をしたままになります。デッドロックしないプログラムを書きなさい。