オペレーティングシステム

Operating System

− 2009 年度版 −

福岡工業大学

情報工学部 情報工学科

柴田望洋

BohYoh Shibata

Fukuoka Institute of Technology

本資料について

▪ 本資料は、2009 年度・福岡工業大学 情報工学部 情報工学科２年生の講義

『オペレーティングシステム』

の補助テキストとして、福岡工業大学 情報工学部 情報工学科 柴田望洋が編んだものです。
自分で作るテキスト兼ノートとして活用しましょう。

▪ 参考文献・引用文献等は、資料の最後にまとめて示します。

▪ 諸君が本資料をファイルに綴じやすいように、研究室の学生達が時間を割いて、わざわざ穴を開けるという作業を行っています（一度のパンチで開けることのできる枚数は限られており、気の遠くなるような時間がかかっています）。
必ずＢ５のバインダーを用意して、きちんと綴じましょう。

▪ 本講義では、以下に示すホームページ上の、各ドキュメントも参照します。
柴田望洋後援会オフィシャルホームページ　http://www.BohYoh.com/

▪ 本資料は、Adobe 社の DTP ソフトウェア InDesign CS4 を用いて作成しています。

オペレーティングシステムとは

まずは、広義のオペレーティングシステムを理解しましょう。

オペレーティングシステムと基本ソフトウェア

広義の**オペレーティングシステム**（*operating system*）すなわち**基本ソフトウェア**（*basic software*）は、主に以下の三つから構成されるソフトウェアです。

・制御プログラム（狭義のオペレーティングシステム）
・汎用言語プロセッサ
・サービスプログラム（ユーティリティプログラム）

■ 制御プログラム（control program）

ジョブ管理、タスク管理、データ管理、入出力管理などを行い、コンピュータの資源を効率よく利用するためのソフトウェアです。

■ 言語プロセッサ（language processor）

プログラムの翻訳などを行います。

■ サービスプログラム（service program）

ファイルのコピー、ソート（整列）、連係編集プログラムなど、OSで提供されるユーティリティソフトウェアです。

■第２種　平成７年度（1999 年度）秋期 午前・問 35

次の文は、言語プロセッサ、サービスプログラム及び制御プログラムについて記述したものである。a、b、cの組合せとして正しいものはどれか。

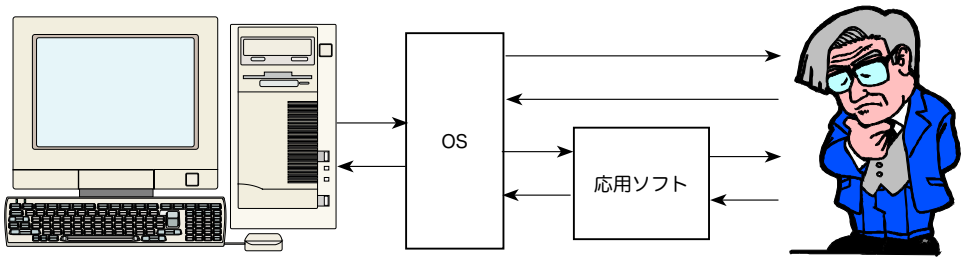
a　資源割振り、スケジューリング、入出力制御、データ管理などを行う。
b　プログラムの翻訳などを行う。
c　整列プログラム、診断プログラム、追跡プログラムなど、一般的に必要とされるプログラムを提供する。

	a	b	c
ア	言語プロセッサ	サービスプログラム	制御プログラム
イ	サービスプログラム	言語プロセッサ	制御プログラム
ウ	サービスプログラム	制御プログラム	言語プロセッサ
エ	制御プログラム	言語プロセッサ	サービスプログラム
オ	制御プログラム	サービスプログラム	言語プロセッサ

オペレーティングシステムとは

コンピュータを利用する際には、必ず何らかのソフトウェアが動作しています。そのソフトこそが、**オペレーティングシステム**（OS：*operating system*）です。

> OS とは計算機を利用するために必要不可欠な一種の基本的なソフトウェアであり、**リソース**（resource ／資源）を管理し、よりよい環境を利用者に提供する。



■各種の OS

▪ UNIX

ワークステーション用の代表的な OS であり、マルチユーザ・マルチタスクをサポートします。

AT&T（American Telephone & Telegraph）社のベル研究所の研究員だった Ken Thompson が、ミニコンピュータ DEC PDP-7 を使いこなすために開発し始めたのがきっかけとなって作られた OS で 1970 年頃に完成しました。開発には Dennis Ritchie が加わり、C 言語を産み出すことになりました。

彼らは、MIT の Multics プロジェクトに参加していたため、UNIX は Multics の影響を大きく受けています（多くのことを目指すよりも、一つのことを目指したため、Multics の［multi ＝複］を［uni ＝単］に置き換えた unics に由来するネーミングが行われました）。

1975 年、UNIX は大学などの研究所へ配布が開始され、さらに 1978 年には、C コンパイラを含むようになり、世界中に普及することになりました。1983 年には、Dennis Ritchie と Ken Tompson は、UNIX オペレーティングシステムの設計と開発という功績により、チューリング賞を受賞しています。

技術的な特徴は以下のとおりです。

・入出力装置などのデバイスを、ファイルと同様に、ストリームとして扱うことができる。
・いろいろなコマンドインタプリタ（文字によるコマンドによって対話的に処理を行うことができます）が提供される。
・ファイルを相対パス・絶対パスで指定できる。
・仕様が公開されて移植性が高いので、広範囲の機種で利用できる。

▪ Microsoft Windows（Vista／XP／2000／NT／Me／98 など）

パソコン用の代表的な OS です。シングルユーザ・マルチタスクをサポートします。

現在、世界でもっとも有名なソフトウェア会社となっている Microsoft 社が開発（もともと某会社が作っていたものを買収）した OS である MS-DOS から発展した OS です。いまや世界中に広がっており、その勢いが止まることはないのかもしれません。

元会長のビル・ゲイツ氏は、高校生の頃から全米的に有名になってしまった、（元）パソコン坊やです。世界中のソフト業界が、この坊やの思いつきに左右されているといっても過言ではありません。この坊やの武勇伝などは、いろいろな伝書などで紹介されているので、ソフト業界を目指す人は、読んでおくとよいでしょう。

▶ MS–Windows98 の前身である MS–Windows 3.1 は、MS–DOS という OS 上で動作するソフトウェアであり、OS そのものではありませんでした。また現在主流となっている MS–Windows XP は、Wiondows 95 や 98 ではなくて、Windows NT および Windows 2000 の後継です。

■第２種　平成８年度（2000 年度）春期 午前・問 35

オペレーティングシステム（OS）の一つである UNIX の説明として、適切でないものはどれか。

ア　キャラクタベースのコマンドを用いた対話方式のヒューマンインタフェースを提供する。
イ　仕様が公開されており移植性が高いので、広範囲な機種で採用されている。
ウ　シングルユーザ、マルチタスク OS である。
エ　分散処理を容易に実現するネットワーク機能を提供する。
オ　ワークステーションの代表的な OS である。

■第２種　平成９年度（2001 年度）春期 午前・問 32

UNIX に関する記述として、正しいものはどれか。

ア　コマンドインタプリタは 1 種類に統一されている。
イ　シングルユーザ、マルチプロセスシステムである。
ウ　入出力装置をファイルと同等に扱うことができる。
エ　ファイルへのパスの表記は 1 通りである。

UNIX とC言語 [1)]

1969 年、大掛かりな Multics プロジェクトは、失敗に終わろうとしていた。目標としていた、高速で使いやすいオンラインシステムはおろか、使えるだけというものすらまったくできあがっていなかったのだ。最後には、何とか動く Multics を作り上げたのは確かだ。しかし IBM が OS/360 で陥ったのと同じ罠に、彼らもやはり引っかかったのだ。彼らが作ろうとした OS はあまりにも巨大で、それに引き換え、使えるハードウエアはあまりにも貧弱だった。Multics は、解決しなくてはならない技術的問題の宝庫だっただけでなく、「小さいことは美しい」という教訓を C に残してくれたのだ。

幻滅と共に Multics プロジェクトから手を引いたベル研の開発者たちは、次に取り組むべき課題を探していた。その一人 Ken Thompson は、別の OS を作成することに熱意を燃やし、何通もの提案書を上司に提出していた（片端からボツになったけど）。承認が出るのを待つ間に、Thompson とその同僚の Dennis Ritchie は、Thompson が作った“Space Travel”というゲームソフトをほったらかしにされていた PDP-7 というコンピュータに移植して楽しんでいた。Space Travel は太陽系を大雑把にシミュレートしており、グラフィック画面上で宇宙船を飛ばし、いろいろな惑星へ着陸することができた。そのために Thompson は猛烈な努力をして、Multics よりはるかに単純で軽快な OS の基礎を PDP-7 に搭載したのだ。OS はすべてアセンブラで書かれていた。1970 年、Brian Kernighan はこの OS を“UNIX”と命名した。もちろんこれは、Multics から学んだ教訓、何をしてはならないかをもじったものだ。

そう、卵と鶏のどちらが先かという点では、間違いなく UNIX が C よりも先に誕生したのである（UNIX の時間情報が、1970 年 1 月 1 日からの通算秒で表されているのもこのためだ － ここからすべてが始まったのである）。アセンブラを使ったことは、困った問題も引き起こした。コードを書くのは骨の折れる仕事だし、デバッグや読むのにも手間がかかる。Thompson は高級言語を使うメリットに惹かれていた。ただし Multics プロジェクトで体験した、PL/I の性能や複雑さはもう御免だった。Thompson はちょっとの間 FORTRAN を試してみたがうまく行かず、B という新しい言語を作成した。これは研究として作られた BCPL という言語を、PDP-7 の 8K ワードのメモリにインタプリタが載るように単純化したものだ。B は決して成功を収めた言語ではない。ハードウエアのメモリ量の制約から、コンパイラではなくインタプリタしか用意できなかった。そのため UNIX 本体を記述するには、B の性能はあまりにも低すぎたのだ。

1970 年になって、新しいマシン PDP-11 への切り替えが行われると共に、型なし言語を開発に使うのには無理があることが明らかになった。性能的な問題もあり、Thompson は PDP-11 用の OS の開発に B 言語を使わず、今回もアセンブラで書くことにした。Dennis Ritchie は PDP-11 の上で、型をサポートし、かつ高性能な“New B”を作ることにした。“New B” －名前はすぐに“C”に決まった － はインタープリタではなくコンパイラであり、型を導入し、変数を宣言してから使うという仕様になっていた

オペレーティングシステムの主な機能

オペレーティングシステムの主要な機能を以下に示します。

① ジョブ管理（*job management*）

ジョブのスケジュール管理、ハードウェア資源の割当てなどを行います。

② タスク管理（*task management*）

タスクの実行を監視して、CPU の割当てや主記憶など、ジョブに与えられる資源の管理などを行います。

③ データ管理（*data management*）

データを主記憶に割り付けたり、磁気ディスクへのファイルのアロケーション、ディレクトリの管理などを行います。

④ 記憶管理（*memory management*）

主記憶を効率よく活用するために、実記憶管理・仮想記憶管理などを行います。

⑤ 運用管理（*operation management*）

稼動状況の報告など運用業務を支援します。

⑥ 障害管理（*fault management*）

障害処理を行い、信頼性や安全性を向上させます。

⑦ 入出力管理（*input-output management*）

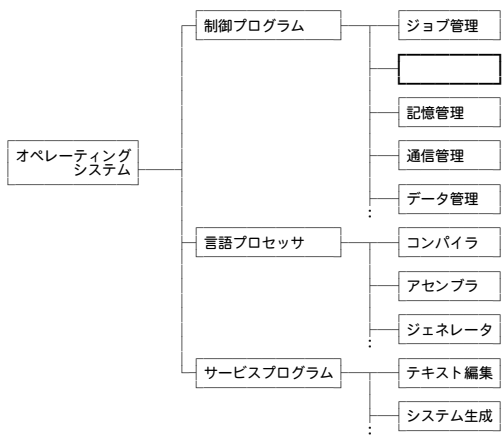
各入出力装置の制御を円滑に行います。

⑧ 通信管理（*communication management*）

複数のコンピュータ間の通信を制御します。

■第２種　平成６年度（1994 年度）秋期 午前・問 26

次の図は、オペレーティングシステムの機能の一部を体系化したものである。［　　］に入れるべき適切な機能はどれか。



ア　オーバーレイ管理　　イ　カタログ管理　　ウ　タスク管理
エ　プロジェクト管理　　オ　メッセージ管理

基本情報技術者試験について

就職活動をする際に、コンピュータ関連の資格をもっていると有利になる（少なくとも不利にならない）のは確かです。在学中にぜひ資格をとりましょう。

▶ 就職後に、会社に勤務しながら資格を取るのは、時間的な制約などの点から、かなり困難であることを知っておきましょう。

*

私が大学院博士後期課程に在学していたとき、知人の紹介で公立の専門学校で講師のアルバイトをすることになりました。時給（50 分あたりの給料）は、次のようになっていました。

大学教授 … ¥5,500　　大学助教授 … ¥4,500　　大学生 … ¥2,500

しかし、給与規定は資格などによっても細分化されており、第２種情報処理技術者（現在の基本情報技術者）取得者の時給は¥3,500 で、私がもっていた資格である第１種情報処理技術者（現在のソフトウェア開発技術者）取得者は¥4,500 でした！

合格に対する一時金や、資格に対する手当て（毎月の給料に¥3,000 ～ ¥9,000 がプラスされる！）が支給される会社もあります。

*

基本情報技術者試験は、春と秋、年２回行われます。詳しくは、以下のホームページを参照してみてください。

http://www.jitec.jp/

OS とミドルウェア

ここでは、ミドルウェアについて学習します。

ミドルウェア

ミドルウェア（*middleware*）とは、基本ソフトウェアと応用ソフトウェアの間に位置して、多数の利用者が共通に使用する基本処理機能を提供するソフトウェアのことです。

異なるメーカの機種間での応用ソフトウェアの移植性の確保や、相互接続を容易にするために導入されます。代表的なものとして、以下のようなものがあげられます。

- データベース管理システム
- 通信管理システム
- ソフトウェア開発支援ツール
- 日本語ワードプロセッサ
- 表計算ソフト
- グラフィック処理システム

ソフトウェアの分類を以下に示します。

ソフトウェア
- ■システムソフトウェア
  - □基本ソフトウェア
  - □ミドルウェア
- ■応用ソフトウェア
  - □個別応用ソフトウェア
  - □共通応用ソフトウェア

■第２種　平成 10 年度（1998 年度）秋期 午前・問 43

異なるメーカの機種間での応用ソフトウェアの移植性の確保や、相互接続を容易にするために導入されるソフトウェアであり、システムの構成を次のように 4 階層で表現した場合に、［　　］に位置付けられるものはどれか。

応用ソフトウェア
a
基本ソフトウェア
ハードウェア

ア　グループウェア　　イ　シェアウェア
ウ　ファームウェア　　エ　ミドルウェア

情報処理システムの基本

OS と関連深い事項として、情報処理システムについて簡単に学習しましょう。情報処理システムは、以下のように発展してきました。

（1）バッチ処理からオンライン処理へ
（2）集中処理→分散処理→エンドユーザコンピューティング
（3）定量的情報処理→定性的情報処理（マルチメディア化を含む）

バッチ処理方式（batch processing）

ユーザからの仕事を蓄えておき、都合の良いときに（あるいは一定の周期で）逐次処理を行っていく方式です。

ジョブ（job）
プログラム実行の一連の作業

入力	処理	出力
カードリーダ（キーボード）	→ プログラム →	プリンタ（ディスプレイ）

バッチ処理方式では、ジョブをまとめて逐次処理します。

A → B → C

利用者は、プログラムやデータなどをセンターに提出し、センターでは、一定期間分のジョブを一括処理します。ジョブを実行開始することを、「ジョブを投入する」といい、その開始・実行などを制御する言語を**ジョブ制御言語**（JCL：*job control language*）と呼びます。

特徴
（1）オペレーティングシステムの働きにより、ジョブの切換え作業を自動化できるので、コンピュータの利用効率を上げることができる。

（2）ユーザに結果が戻るまでの応答時間＝ターンアラウンドタイムが長くなる。

センターバッチ処理（*center batch processing*）

↓

リモートバッチ処理（*remote batch processing*）

入出力装置をユーザの近く（センターにとっては遠隔地）に設置して、ターンアラウンドタイムを縮小する。

（3）入出力には、時間がかかるため、その間に CPU が遊んでしまう。これをアイドル時間と呼ぶ。

例】給与計算、売り上げ集計（日次、月次、年次）、成績集計、
大規模な科学技術計算…

■第２種　平成 7 年度（1995 年度）春期 午前・問 55

次の条件を満たすコンピュータの処理形態として、最も適切なものはどれか。

・入出力はコンピュータの設置場所と離れた遠隔装置から行う。
・あらかじめ蓄えられたデータをまとめて処理する。

ア　オンラインリアルタイム処理　　イ　対話型処理
ウ　バッチ処理　　エ　リモートバッチ処理
オ　ロールバック処理

■第２種　平成 12 年度（2000 年度）秋期 午前・問 33

あるジョブのターンアラウンドタイムを解析したところ、1,350 秒のうち CPU 時間が 2／3 であり、残りは入出力時間であった。1 年後に、CPU 時間はデータ量の増加を考慮しても、性能改善によって当年比 80%に、入出力時間はデータ量の増加によって当年比 120%になることが予想されるとき、このジョブのターンアラウンドタイムは何秒になるか。ここで、待ち時間、オーバヘッドなどは無視する。

ア　1,095　　イ　1,260　　ウ　1,500　　エ　1,665

オンラインシステム（on–line system）

ユーザからの要求（データの発生等）に対して即座に処理を行う方式です。

遠隔地において、要求の発生と同時に処理をする方式であり、**リアルタイムシステム**（*real-time system*）として実現されます。応答時間＝レスポンスタイムが短いことが要求されます。

特徴

（１）要求の発生に対して即座に処理が行われ、応答時間が短い。

（２）座席の予約状況、口座の預金状況などの必要な情報は常に更新される。

※そのために、システムが複雑・高価になりやすい。

（３）システムの運営、とくに信頼性、保安性などの維持などが容易ではない。

例】座席予約システム、
銀行のオンラインシステムＡＴＭ（*Automatic Teller Machine*：自動預金機）
プロセス制御、交通管制システムなど

二つの応答時間

バ ッ チ 処理方式の応答時間 → ターンアラウンドタイム
オンライン処理方式の応答時間 → レスポンスタイム

ターンアラウンドタイム（*turnaround time*）

直訳すると往復所要時間です。ジョブを投入してから、最終的な処理結果が出力されるまでの時間のことです。

レスポンスタイム（*response time*）

利用者が問合せ又は要求終了を端末で指示してから、その端末に処理結果の出力が始まるまでの時間のことです。

シューティングゲームにおいて、キャラクタを移動するキーボード操作に対して、すぐにそのキャラクタは移動すべきです。もしレスポンスタイムが長ければ、キーボードを操作してから、キャラクタが移動するまでにタイムラグが発生し、使いにくいものとなってしまいます。

■第２種　平成６年度（1994 年度）秋期 午前・問 65

オンラインシステムにおいて、利用者が問合せ又は要求終了を端末で指示してから、その端末に処理結果の出力が始まるまでの時間を何というか。

ア　アクセス時間　　イ　応答時間　　ウ　ターンアラウンドタイム
エ　プロセス時間　　オ　リアルタイム

■第２種　平成７年度（1995 年度）秋期 午前・問 52

コンピュータシステムに対して問合せ又は要求の終わりを指示してから、利用者端末に最初の応答が出始めるまでの時間を何というか。

ア　アクセスタイム　　イ　サイクルタイム　　ウ　ターンアラウンドタイム
エ　リアルタイム　　オ　レスポンスタイム

■第２種　平成９年度（1997 年度）春期 午前・問 52

オンライン検索システムにおいて、利用者が検索コマンドの送信を端末で指示してから、その端末に検索結果の最初の出力が始まるまでの時間を何というか。

ア　アクセス時間　　イ　位置決め時間
ウ　応答時間　　エ　ターンアラウンドタイム

■第２種　平成 11 年度（2001 年度）秋期 午前・問 52

コンピュータシステムに対して問合せ又は要求の終わりを指示してから、利用者端末に最初の処理結果のメッセージが出始めるまでの時間を何というか。

ア　アクセスタイム　　イ　サイクルタイム
ウ　ターンアラウンドタイム　　エ　レスポンスタイム

TSS =タイムシェアリングシステム（time-sharing system）

短い時間間隔で、ユーザを切換えてサービスを行う方式。

オンラインシステムでは、データ等は遠隔地から送ることができるが、実際のジョブはセンターで行われる。

↓

ジョブ自身を、利用者が行いたい。すなわち、複数の人間が端末において、あたかも自分が計算機を占有しているかのように使いたい。

コンピュータが、短い時間間隔で利用者を切り換えて会話的な（interactive）サービスを行う方式。

特徴

（1）対話型処理が可能。
（2）ユーザはコンピュータを占有しているかのようにプログラムの実行が可能。
（3）コンピュータの持つ機能を共同利用できる。

例】 大学、共同利用施設、文献検索など

■第２種　平成 10 年度（1998 年度）秋期 午前・問 54

次の三つの業務と、それらの処理形態の最も適切な組合せはどれか。

〔業務〕1. １か月の給与計算
2. 工業用ロボットの自動運転
3. 飛行機の座席予約

〔処理形態〕A. オンライントランザクション処理
B. バッチ処理
C. リアルタイム処理

	1	2	3
ア	A	B	C
イ	A	C	B
ウ	B	C	A
エ	C	A	B

分散処理方式（distributed processing）

複数のコンピュータが対等に結合し、情報のやり取りを行うコンピュータネットワーク上において、複数のコンピュータを利用する方式です。

コンピュータネットワーク：

ＬＡＮ（local area network）

ＷＡＮ（wide area network）

分散処理において、処理の大部分は自分の部門のコンピュータで行い、自分のコンピュータで処理しきれないものを、ネットワークを通じて、より高速・大型のコンピュータや、機器などを利用します。

集中処理システムと分散処理システムの長所と短所を比較すると、次のようになります。

	集中処理システム	分散処理システム
長所	■センタに集中して対策を施すことによって、機密保護やセキュリティやデータの一貫性を維持・管理することが容易である。 ■ネットワークの性能はシステム全体の性能にあまり影響を与えない。 ■管理の集中化によって、人件費を節約できる。 ■コストパフォーマンスが低くなる傾向がある。	■個々のシステムの資源管理が容易である。 ■災害や障害の際の損害が局所化される。 ■機能や設備が分散しているのでシステムの変更が容易。 ■安価なシステムを組み合わせることにより、経済的なシステムの構築が容易である。
短所	■一部の装置の故障がシステム全体の停止につながることが多く、災害や障害の際の修復作業は容易ではない。 ■機能の拡張や業務量の増大に対応したシステムの拡張などが困難である。 ■システムの規模が大きくなりやすい。 ■規模が大きくなると、こまわりがきかないため新技術の導入が困難になりやすい。	■システム全体を効率よく運用するための運用管理が複雑になりやすい。 ■異常が発生した際に、原因の特定が困難である。 ■ネットワークにおけるデータの遅延が全体の性能を低下させる。 ■情報の安全管理、末端での管理が困難である。

■第２種　平成７年度（1995 年度）秋期 午前・問 54

集中処理システムと比較した場合の分散処理システムの特徴に関して、正しい記述はどれか。

ア　一部の装置の故障がシステム全体の停止につながることが多い。
イ　機能の拡張や業務量の増大に対応したシステムの拡張などが困難である。
ウ　機密保護やセキュリティの確保が容易である。
エ　システム全体を効率よく運用するための運用管理が複雑になりやすい。
オ　ネットワークの性能はシステム全体の性能にあまり影響を与えない。

■第２種　平成９年度（1997 年度）秋期 午前・問 54

集中処理システムと比較した場合の分散処理システムの特徴に関して、正しい記述はどれか。

ア　一部の装置の故障がシステム全体の停止につながることが多い。
イ　機能の拡張や業務量の増大に対応したシステムの拡張などが困難である。
ウ　機密保護やセキュリティの確保が容易である。
エ　システム全体を効率よく運用するための運用管理が複雑になりやすい。

■第２種　平成 11 年度（1999 年度）秋期 午前・問 55

分散処理システムと比較したとき、集中処理システムの記述として、最も適切なものはどれか。

ア　災害や障害のときにセンタ側で集中した修復作業を行うことができるので、システム全体が長時間停止する危険性を回避できる。
イ　システムを一括管理しているので、システム機能の追加・変更などの要求に応ずることが容易であり、バックログの堆積が起こりにくい。
ウ　センタに集中して対策を施すことによって、セキュリティやデータの一貫性を維持・管理することが容易である。
エ　ハードウェア及びソフトウェア資源の運用・管理が煩雑になるが、新技術に対応した拡張が容易である。

■基本　平成 13 年度（2001 年度）春期 午前・問 37

広範な地域に配置した複数の計算機システムで構成される分散処理システムと単一のセンタで運用される集中処理システムを比較したとき、集中処理システムの特徴として、最も適切なものはどれか。

ア　災害や障害のときにセンタ側で集中した修復作業を行うことができるので、システム全体が長時間停止する危険性を回避できる。
イ　システムを一括管理しているので、システム機能の追加・変更などの要求に応ずることが容易であり，バックログの堆積が起こりにくい。
ウ　センタに集中して対策を施すことによって、データの一貫性を維持・管理しやすい。
エ　ハードウェア及びソフトウェア資源の運用・管理が煩雑になるが、新技術に対応した拡張が容易である。

タスク管理

ここでは、タスク管理について学習します。

コンピュータ内部での処理形態

ますば、コンピュータ内部での処理形態の概略を理解しましょう。

1．ユニプロセッサ・ユニプログラミング方式（uniprocessor–uniprogramming）

1 台の CPU を有するコンピュータが、1 個のプログラムを処理します。

2．マルチ（多重）プログラミング方式（multiprogramming）

主記憶上に複数のプログラムを格納し、プログラムを切り替えることによって、コンピュータの利用効率を向上させる方式。最近はパソコンでもサポートされています。

※実際に同時に動作するプログラムは 1 個です。



3．マルチプロセッサ方式（multiprocessor）

複数の CPU を用いて、マルチプログラミングをサポートする方式です。

※ CPU の数のプログラムが同時に動作します。





タスク管理の目的

タスク（task）とは、オペレーティングシステムから見たときの、ジョブの制御単位のことです。

タスク管理の主目的は、CPU を効率よく利用することです。

現在のコンピュータでは、主記憶上に複数のプログラムを格納しておき、プログラムを切りかえることによって、コンピュータの利用効率を向上させる方式である**マルチプログラミング**（**多重プログラミング**あるいは**マルチタスキング**とも呼ばれる）が行われます。

そのようなシステムでは、CPU が、あるプログラムから他のプログラムへと非常に短い時間で切り替えられ実行される。ある瞬間には単一のプログラムのみが実行されているのですが（CPU が 1 個の場合）、秒単位でみると、複数のプログラムが同時に実行されているかのように見えます。

■ プロセス（process）

コンピュータのユーザは、計算機の中に入り込むことはできません。OS の初期の時代では、“実行中のプログラム”をプロセスと考えてきました。現在では、リソースを要求するユーザの代理であると考えられています。プロセスはリソースを要求し、リソースはプロセスによって要求されます。

■ 仮想 CPU（virtual CPU）

各プロセスには、あたかも自分占有の CPU を有して実行しているかのように見えますが、このように各プロセスに対して作られた見せかけの CPU を**仮想 CPU**（*virtual CPU*）と呼びます。プロセスは、CPU の処理時間の一部分しかサービスを受けられないため、仮想 CPU の実行速度は、CPU より遅くなることになります。

+

■ CPU スケジューリング（CPU scheduling）

一般に CPU の数は、全ての実行中のプログラム（プロセス）に割り当てることができるほど十分ではなく、プログラムの間で切り換えて利用する必要があります。

どのように切り換えを行うかを計算して制御することを CPU **スケジューリング**と呼び、それを行うプログラムを **CPU スケジューラ**（CPU scheduler）と呼びます。また、切りかえを行うのが**ディスパッチャ**（*dispatcher*）です。

■第２種　平成７年度（1995 年度）春期 午前・問 33

次に示す機能は、汎用計算機のオペレーティングシステムにおけるジョブ管理、タスク管理、データ管理及び入出力管理のいずれかに含まれる機能である。
この中で、タスク管理に含まれるものはどれか。

ア　ＣＰＵ割当て　　イ　ジョブスケジュール機能　　ウ スプール機能
エ　入出力の実行機能　　オ　ファイルの保護機能

■第２種　平成 10 年度（1998 年度）春期 午前・問 32

オペレーティングシステムのタスク管理に含まれる機能はどれか。

ア　CPU 割当て　　イ　スプール制御　　ウ　入出力の実行　　エ　ファイル保護

■第２種　平成８年度（1996 年度）秋期 午前・問 32

タスク管理の目的として適切なものはどれか。

ア　CPU を効率よく使用する。
イ　運用管理者によって行われる運用管理業務を支援する。
ウ　記憶領域を効果的に使用するとともに、主記憶の容量の制約を緩和する。
エ　ハードウェアを意識しないデータの蓄積、処理、保存、運用を可能にする。
オ　ハードウェアを監視し、運用状況を把握してシステムの維持管理を行う。

■第２種　平成 11 年度（1999 年度）秋期 午前・問 32

OS におけるタスク管理の目的はどれか。

ア　オペレータにコマンド形式の対話インタフェースを提供する。
イ　仮想記憶を効率的に実現する。
ウ　処理装置の利用効率を高めるように制御する。
エ　ハードウェアを意識しないで、データが処理できるようにする。

■第２種　平成９年度（1997 年度）秋期 午前・問 35

オペレーティングシステムのタスク管理の役割として、正しいものはどれか。

ア　各種の補助記憶装置へのアクセス手段を、装置に依存しない形態で提供し、応用プログラム作成の負担を軽減する。
イ　仮想記憶空間を提供し、実記憶を有効に利用する。
ウ　入出力装置の制御を行い、正確かつ効率よく入出力装置を動作させる。
エ　マルチプログラミングの制御を行い、CPU を有効に利用する。

■第２種　平成 12 年度（2000 年度）春期 午前・問 34

タスク管理の役割として、適切なものはどれか。

ア　各種の補助記憶装置へのアクセス手段を、装置に依存しない形態で提供し、応用プログラム作成の負担を軽減する。
イ　仮想記憶空間を提供し、実記憶を有効に利用する。
ウ　入出力装置の制御を行い、正確かつ効率良く入出力装置を動作させる。
エ　マルチプログラミングの制御を行い、CPU を有効に利用する。

■基本　平成 20 年度（2008 年度）秋期 午前・問 28

タスク管理の役割として、適切なものはどれか。

ア　各種の補助記憶装置へのアクセス手段を、装置に依存しない形態で提供し、応用プログラム作成の負担を軽減する。
イ　仮想記憶空間を提供し、実記憶を有効に利用する。
ウ　入出力装置の制御を行い、正確かつ効率良く入出力装置を動作させる。
エ　マルチプログラミングの制御を行い、CPU を有効に利用する。

■第２種　平成８年度（1996 年度）秋期 午前・問 33

オペレーティングシステムが、CPU 資源を割り当てる対象を示す用語を二つ選べ。

ア　クライアント　　イ　セグメント　　ウ　タスク
エ　プロセス　　オ　ユーザ

■第２種　平成６年度（1994 年度）秋期 午前・問 62

オペレーティングシステムにおいて、あるプログラムの実行中に、入出力など処理装置に待ちが生じたとき、処理装置を直ちに別のプログラムの実行に割り当てる。このようにして、システム全体としての処理効率の向上をねらう方式を何というか。

ア　エミュレーション　　イ　タイムシェアリング　　ウ　多重プログラミング
エ　パイプライン処理　　オ　リアルタイム処理

■第２種　平成８年度（1996 年度）春期 午前・問 31

あるプログラムの実行中に、入出力などのために処理装置が待ち状態になったとき、処理装置を他のプログラムの実行に割り当てることによって処理装置を有効に利用する方式を何というか。

ア　オーバレイ　　イ　スラッシング
ウ　ダイナミックアロケーション　　エ　マルチプログラミング
オ　ラウンドロビン

■第２種　平成 10 年度（1998 年度）秋期 午前・問 35

あるプログラムの実行中に、入出力などのために処理装置が待ち状態になったとき、処理装置を他のプログラムの実行に割り当てることによって処理装置を有効に利用する方式を何というか。

ア　スラッシング　　イ　ダイナミックアロケーション
ウ　マルチプログラミング　　エ　ラウンドロビン

■第２種　平成９年度（1997 年度）春期 午前・問 51

オペレーティングシステムにおいて、あるプログラムを実行中に、入出力などで CPU に待ちが生じたとき、CPU を直ちに別のプログラムに割り当てる。

このような機能を実現するために、オペレーティングシステムがもつべき処理機能を何というか。

ア　タイムシェアリング　　イ　マルチタスキング
ウ　マルチプロセッシング　　エ　リアルタイム処理

■基本　平成 13 年度（2001 年度）春期 午前・問 32

あるプログラムの実行中に、入出力などのために処理装置が待ち状態になったとき、処理装置をほかのプログラムの実行に割り当てることによって有効に利用する方式を何というか。

ア　スラッシング　　イ　ダイナミックアロケーション
ウ　マルチプログラミング　　エ　ラウンドロビン

■第２種　平成 11 年度（1999 年度）秋期 午前・問 34

多重プログラミングに関する記述として、適切なものはどれか。

ア　ジョブとしては多重で処理されるが、シングルタスクで実行する。
イ　タスクの実行中に、入出力などを行ったために生じる CPU の空き時間を利用して、別タスクを並列に実行する。
ウ　プログラムの実行中に、自分自身を呼び出して実行するプログラミング方法である。
エ　プログラムを並列に処理するので、ハードウェアとして複数のプロセッサとメモリを結合した並列処理システムが必要である。

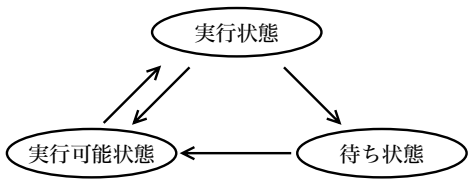
■基本　平成 13 年度（2001 年度）春期 午前・問 33

OS の機能の中で、実行可能状態にあるプロセスの中から次に実行すべきプロセスに CPU の使用権を与えて、実行状態にするものはどれか。

ア　アロケータ　　イ　イニシエータ　　ウ　ターミネータ　　エ　ディスパッチャ

プロセスの状態遷移

以下に示すのは、プロセスの状態遷移図です。



■状態

・実行（running）

CPU を確保し、プログラムを実行している状態です。

・実行可能（ready）

CPU が割り当てられればすぐプログラムを実行できる状態です。

・待機（wait）

要求しているリソース、たとえば入出力機器が、空くのを待っているとか、メッセージが到達するのを待っているとか、何らかの待ち状態にあり、CPU を必要としていない状態です。

■状態遷移

・実行可能状態から実行状態へ

CPU スケジューラ（CPU scheduler）すなわちディスパッチャによって、CPU を割り当てられたときに生じます。

・実行状態から実行可能状態へ

プリエンプション（より優先度の高いジョブに割り込まれることによって実行が中断される）によって、あるいは自分に割り当てられた時間＝クオンタム（quantum）を使い果たしたときに生じます。

・実行状態から待ち状態へ

プロセスが入出力動作を起動し、その完了を待たなくなければいけなくなったとき、何らかのリソースが必要となり、それが空くのを待たなければならなくなったときに生じます。他の遷移と異なり、プロセス自らの操作によって生じます。

・待機状態から実行可能状態へ

入出力動作が完了したり、他のプロセスの命令により待ち状態が解除されたときに生じます。

■第２種　平成８年度（1996 年度）春期 午前・問 32

タスクの状態に関する次の記述中の［　　］に入れるべき適切な字句の組合せはどれか。

タスクは三つの状態で管理される。これらは、CPU 使用権を与えられた［ a ］状態、CPU 使用権を持っている［ b ］状態、及び入出力要求などをきっかけに［ a ］状態から移行する［ c ］状態である。

	a	b	c
ア	受付け	待ち	実行
イ	実行	実行可能	待ち
ウ	実行	待ち	割込み
エ	実行可能	受付け	割込み
オ	実行可能	割込み	実行

■第２種　平成 12 年度（2000 年度）春期 午前・問 33

プロセスは、実行可能状態（ready）、実行状態（running）、待ち状態（wait）を遷移しながら実行される。プロセスの状態遷移に関する記述のうち、適切なものはどれか。

ア　CPU 処理と入出力処理が交互に現れるプロセスを複数個同時に実行させると、各プロセスは実行状態と待ち状態の二つの状態間だけを遷移する。
イ　実行可能状態とは、CPU の割当てを待っている状態をいう。実行可能状態のプロセスは一般に複数個存在し、これらは待ち行列を形成する。
ウ　時分割処理を行っているシステムでは、実行状態のプロセスは、一定時間が経過すると、待ち状態に遷移する。
エ　マルチプログラミングシステムでは、CPU が 1 個でも、実行状態のプロセスは複数個存在する。

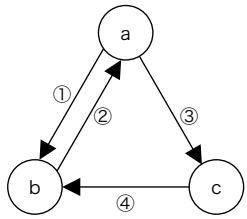
■第１種　平成８年度（1996 年度）午前・問 23

タスクは、実行可能状態（ready）、実行状態（running）、待ち状態（wait）の三つの状態を繰り返しながら実行される。タスクの状態遷移に関する記述のうち、誤っているものはどれか。

ア　実行可能状態におかれたタスクが、待ち状態に移されることはない。
イ　実行状態のタスクは、ある種の事象の完了（例えば入出力動作の完了）を待つ必要が生じたとき、待ち状態におかれる。
ウ　実行状態のタスクは、自分に割り当てられた CPU 時間が終了すると待ち状態におかれる。
エ　タスクの実行が完了すると、そのタスクが使用していた資源は解放され、他のタスクが使用できるようになる。
オ　待ち状態のタスクは、待ちの原因となった事象が完了すると、実行可能状態に移る。

■第２種　平成７年度（1995 年度）秋期 午前・問 39

図はプロセスの状態と遷移を表している。ａ、ｂ、ｃの状態を表しているのはどれか。



状態遷移の要因
①実行優先度の高いプロセスに CPU 使用権が移された。
② CPU 使用権が与えられた。
③入出力動作の完了を待つ。
④入出力動作が完了した。

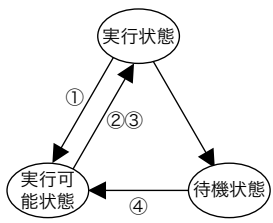
	a	b	c
ア	実行可能状態	実行状態	待機状態
イ	実行可能状態	実行状態	実行状態
ウ	実行状態	実行可能状態	待機状態
エ	実行状態	待機状態	実行可能状態
オ	待機状態	実行状態	実行可能状態

■第２種　平成 11 年度（1999 年度）秋期 午前・問 33

省略（前問と同じ）

■第２種　平成９年度（1997 年度）秋期 午前・問 38

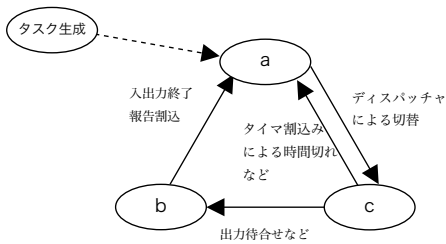
図はプロセスの状態と遷移を表している。状態遷移の要因①～④の正しい組み合せはどれか。



	①	②	③	④
ア	CPU 使用権が与えられた。	実行優先度の高いプロセスに CPU 使用権が移された。	入出力動作が完了した。	入出力動作の完了を待つ。
イ	実行優先度の高いプロセスに CPU 使用権が移された。	CPU 使用権が与えられた。	入出力動作の完了を待つ。	入出力動作が完了した。
ウ	入出力動作が完了した。	入出力動作の完了を待つ。	CPU 使用権が与えられた。	実行優先度の高いプロセスに CPU 使用権が移された。
エ	入出力動作の完了を待つ。	入出力動作が完了した。	実行優先度の高いプロセスに CPU 使用権が移された。	CPU 使用権が与えられた。

■第１種　平成７年度（1995 年度）午前・問 27

図はオペレーティングシステムのタスクの状態遷移を表す。ａ，ｂ，ｃの各状態に当てはまる組合せとして、正しいものはどれか。



	a	b	c
ア	実行可能状態	実行状態	待ち状態
イ	実行可能状態	待ち状態	実行状態
ウ	実行状態	実行可能状態	待ち状態
エ	実行状態	待ち状態	実行可能状態
オ	待ち状態	実行状態	実行状態

■第２種　平成 10 年度（1998 年度）春期 午前・問 33

プロセスの起動から終了までの状態遷移として、あり得ないものはどれか。

ア　起動→実行可能状態→実行状態→終了
イ　起動→実行可能状態→待機状態→実行状態→終了
ウ　起動→実行可能状態→実行状態→実行可能状態→実行状態→終了
エ　起動→実行可能状態→実行状態→待機状態→実行可能状態→実行状態→終了

□第１種　平成９年度（1997 年度）午前・問 26

プロセスの状態は、実行、実行可能、待機の三つを基本と考えることができる。実行状態にあるプロセスがプリエンプションによって他のプロセスに実行を中断され、再び実行状態に戻るまでの状態遷移を表したものはどれか。

ア　実行→実行可能→実行
イ　実行→実行可能→待機　　→実行
ウ　実行→待機　　→実行
エ　実行→待機　　→実行可能→実行

プロセスの切替えの原理

複数のプロセスを見かけ上同時に動かすためには、時間の流れを区切り、それぞれのプロセスにCPUを割り当てなければなりません。

プロセスの切替え方式としては、大きく二つの方法があります。

❶ イベントドリブン（事象駆動 event driven）

事象すなわち**イベント**（*event*）が発生したのを契機にして、CPUを切り換えてタスクのスケジューリングを実行する方式です。マルチプログラミングを実現するための基本となる技術です。

イベントが発生するタイミングの例を以下に示します。

（ａ）入出力の完了
（ｂ）入出力要求の発生
（ｃ）ジョブの到着
（ｄ）ジョブの完了

イベントが生じたときに、実行中のプロセスを一時停止し、そのプロセスの再続行が可能となるようにCPUの状態を保持する**割込み**（*interrupt*）機構が必要となります。

❷ タイムスライシング（時分割 time–slicing）

システムの状態変化とは無関係に、設定した短い時間＝**タイムクオンタム**（*time-quantum*）の周期でプロセスを切り換える方式です。TSSの実現には必要不可欠な技術です。

クオンタムが終了したら割込を生じさせるインタバルタイマが必要となります。

マルチプログラミングの実現は、イベントドリブン方式でのみ可能であり、TSSの実現にはタイムスライシングの機能が必要です。多くのシステムは、マルチプログラミングとタイムシェアリングの両方のサービスを備えており、これら２方式を組み合わせて利用します。

なお、タイムスライシングの導入は、マルチプログラミングシステムの性能を向上させ、イベントドリブンの導入は、タイムシェアリングシステムのリソース利用効率を向上させることが分かっています。

■基本　平成 16 年度（2004 年度）春期 午前・問 31

CPU の処理時間を微小時間に分割し、それを実行可能な状態にあるタスクに割り当てることを何というか。

ア　オーバレイ　　イ　スワッピング
ウ　タイムスライシング　　エ　リアルタイム処理

■第２種　平成９年度（1997 年度）秋期 午前・問 34

CPU の処理時間を微小時間に分割し、それを実行可能な状態にあるタスクに割り当てる形態はどれか。

ア　オーバレイ　　イ　スワッピング
ウ　タイムスライシング　　エ　リアルタイム処理

□第１種　平成８年度（1996 年度）午前・問 28

イベントドリブンスケジューリングに関する記述のうち、正しいものはどれか。

ア　環境の変化をトリガとして、タスクのスケジューリングを実行する。
イ　タスクの優先順位をダイナミックに変更しながらスケジューリングをする方式である。
ウ　タスクを一定周期で実行するためのスケジューリング方式である。
エ　優先度の低いタスクのスケジューリングに適用する。
オ　ラウンドロビン方式と組み合わせて使うことができないスケジューリング方式である。

□第１種　平成 10 年度（1998 年度）午前・問 24

イベントドリブン方式のプロセス切替えに関する記述のうち、正しいものはどれか。

ア　イベントドリブン方式とタイムスライシング方式は排他的なものであり、一つのシステムで共存することはない。
イ　プロセス切替えのきっかけは、入出力終了やプログラム異常などであり、ユーザプログラムが要求するものではない。
ウ　プロセス切替えの処理はソフトウェアによって行われ、ハードウェアが介在することはない。
エ　割り込まれたプロセスが、割込み処理終了後に、引き続いて実行される保証はない。

割込み（interrupt）

JIS X0010 01.09 の定義

> **計算機プログラムの実行のような処理の中断であって、その処理に対する外部からの事象に起因し、後でその処理が再開できるような方法で遂行されるもの。**

あるプログラムの実行中に外部からの何らかの事象によってそのプログラムの実行が一時停止され、その事象に対応した別のプログラムが先に実行されることです。割込んで実行されたプログラムが完了すると、元のプログラムの実行が「中断点」から再開されることになります。

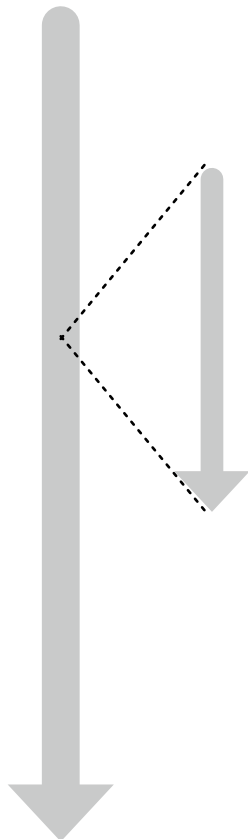
CPU の演算速度は入出力装置の速度に比べて格段に高速の速度差を逆に利用し , あるプログラムの入出力動作の完了を待つことなく、その間に他のプログラムを実行したり、それぞれに優先順位をつけて複数のプログラムを並行処理したりすることでコンピュータシステム全体を有効に動かすために考案された仕組みの一つです。

▶　すべての割込みが無条件に受けつけられるとは限りません。

割込み処理の流れは、以下のようになります。

(1)　ユーザモードから特権モードへの移行
(2)　PSW やレジスタ類の退避
　　※割込み発生時に実行していた命令の次の命令のアドレスが退避されます。
(3)　割込み処理ルーチンの開始番地の決定
(4)　割込み処理ルーチンの実行

割込み処理が完了すると、退避されていたアドレスが復帰され、割込み直前に実行していたプログラムの実行が再開されます。



プログラム状態語ＰＳＷ（Program Status Word）

割込み処理中に、再度の割込みを防止したり、割込み処理後のプログラムの再実行に備えた情報を保存したりする。割込みが発生すると、実行中のプログラムのPSWが待避され、割込み処理終了後に待避されていたPSWが戻される。

割込みは、以下のような条件が成立したときに発生します。

- 入出力動作が完了した
- インタバルタイマの時間が 0 になった
- ハードウェア異常を検出した
- 演算異常（オーバーフロー）を検出した

■命令割込み

・スーパーバイザコール割込み（supervisor call interrupt）

実行モード

特権モード（master mode ／ supervisor mode）
非特権モード（slave mode ／ problem mode）

非特権モードで実行しているプログラムは、オペレーティングシステムの核であるニュークリアス（nucleus）のサービスの要求を SVC 命令によって行います。

■内部割込み（internal interrupt）

・プログラムチェック割込み（program check interrupt）

オーバフロー（演算結果の桁あふれ）、0 による除算、記憶保護違反などによって発生します。

■外部割込み（external interrupt）

・入出力割込み（input/output interrupt）

入出力操作が完了したときや、誤った入出力を指定した場合などに、入出力チャネルにより発生させられます。チャネルが、処理装置に代わって入出力処理を実行することによって、処理装置の動作と入出力動作のオーバーラップ（並行動作）が可能となり、全体の効率が向上します。

・機械チェック割込み（machine check interrupt）／異常割込み

以下に示すような、ハードウェアの障害によって生じます。

・誤動作
・故障
・電源のトラブル　など

この他に、**タイマ割込み**（インタバルタイマの終了）、**外部信号割込み**（他のコンピュータシステムからの信号などによって発生）などがあります。

■基本　平成 14 年度（2002 年度）秋期 午前・問 18

割込み処理の流れを示す次の記述中の［　　］に入る処理はどれか。

〔割込み処理の流れ〕
(1)　ユーザモードから特権モードへの移行
(2)　［　　］
(3)　割込み処理ルーチンの開始番地の決定
(4)　割込み処理ルーチンの実行

ア　CCW（Channel Command Word）の読出し
イ　オペランドの読出し
ウ　資源の割当て
エ　レジスタ類の退避

■基本　平成 14 年度（2002 年度）春期 午前・問 22

割込みが発生すると、あるアドレスが退避され、割込み処理が実行される。割込み処理が完了すると、退避されていたアドレスが復帰され、割込み直前に実行していたプログラムの実行が再開される。退避されていたアドレスはどれか。

ア　割込みが発生したときに実行していた命令のアドレス
イ　割込みが発生したときに実行していた命令の次の命令のアドレス
ウ　割込み処理の最後の命令のアドレス
エ　割込み処理の先頭の命令のアドレス

■第２種　平成７年度（1995 年度）秋期 午前・問 25

割込みに関して、正しい記述はどれか。

ア　タイマ割込みは、内部割込みの一種である。
イ　入出力割込みは、制御装置と入出力装置を並行動作させるために使われる。
ウ　割込みが発生すると、次の割込みまで、制御装置は停止する。
エ　割込みは、実行順序を強制的に変更する分岐命令である。
オ　割込みは、ハードウェアによって発生するので、ソフトウェアで発生させることはできない。

■第２種　平成 10 年度（1998 年度）秋期 午前・問 34

割込みに関する記述として、正しいものはどれか。

ア　入出力割込みによって、処理装置と入出力装置の並行動作が可能となる。
イ　割込みが発生すると、次の割込みまで処理装置は停止する。
ウ　割込みは、その原因によらず無条件に受け付けられる。
エ　割込みは、ハードウェアによって発生するものであり、ソフトウェアでは発生させることができない。

■第２種　平成７年度（1995 年度）春期 午前・問 23

入出力に関する次の記述中の［　　］に入れるべき適切な字句の組合せはどれか。

［ a ］が処理装置に代わって入出力処理を実行することによって、処理装置の動作と入出力動作のオーバーラップが可能となり、全体の効率が向上する。

一方、入出力の終了は［ b ］機能によって処理装置に伝えられ、処理装置の動作と入出力動作の同期がとられる。

	a	b
ア	制御装置	機械チェック割込み
イ	制御装置	入出力割込み
ウ	制御装置	プログラム割込み
エ	チャネル	機械チェック割込み
オ	チャネル	入出力割込み

■基本　平成 15 年度（2003 年度）秋期 午前・問 21

プロセッサの割込みで、外部割込みに分類されるものはどれか。

ア　演算例外　　イ　タイマ
ウ　ページフォールト　　エ　命令コード異常

■基本　平成 13 年度（2001 年度）秋期 午前・問 21

内部割込みに分類されるものはどれか。

ア　記憶保護例外　　イ　タイマ通知　　ウ　電源異常　　エ　入出力動作終了

■第２種　平成 11 年度（1999 年度）春期 午前・問 20

内部割込みの原因となるものはどれか。

ア　コンピュータの電源装置における異常の発生
イ　処理装置内で時間計測を行うカウンタの所定値越え
ウ　入出力装置の動作終了や障害発生

エ　浮動小数点演算でのけたあふれ（オーバフロー）の発生

■基本　平成 15 年度（2003 年度）春期 午前・問 21

プロセッサが割込みを発生するのはどの場合か。

ア　インタリーブ方式によるメモリバンクの切替え完了
イ　キャッシュメモリに対するヒットミスの発生
ウ　入出力開始命令の実行
エ　浮動小数点演算命令実行によるあふれ（オーバフロー）の発生

■第２種　平成９年度（1997 年度）春期 午前・問 29

プログラム割込みの原因となる得るものはどれか。

ア　入出力動作が終了した。
イ　ハードウェアが故障した。
ウ　プログラムで演算結果があふれた（オーバーフローした）。
エ　プログラムの実行時間が設定時間を超過した。

■第２種　平成 12 年度（2000 年度）秋期 午前・問 27

プログラム割込みの原因となり得るものはどれか。

ア　入出力動作が終了した。
イ　ハードウェアが故障した。
ウ　プログラムで演算結果があふれた（オーバフローした）。
エ　プログラムの実行時間が設定時間を超過した。

■割込みの優先順位

割込みには優先順位があります。その概略を以下に示します。

優先順位	割込みの原因
1	機械チェック
2	外部割込み
3	入出力割込み（入出力処理の終了）
4	ＳＶＣ／プログラム割込み

■多重割込み

割込み処理の実行中に割込みが発生することを**多重割込み**といいます。ただし、高い優先順位の割込み処理の実行中は、優先順位の低い割込みはマスクされます。

□第１種　平成８年度（1996 年度）午前・問 21

割込みに関する記述のうち、正しいものはどれか。

ア　割込み処理が、更に割り込まれることはない。
イ　割込み処理は、先に発生した割込みから順番に行われる。
ウ　割込みは禁止することができる。
エ　割込みは、入出力の終了を知らせるためだけに用いられる。
オ　割込みを明示的に発生させるための命令はない。

□第１種　平成 10 年度（1998 年度）午前・問 98

リアルタイム OS が行う多重割込みの処理方法として、適切なものはどれか。

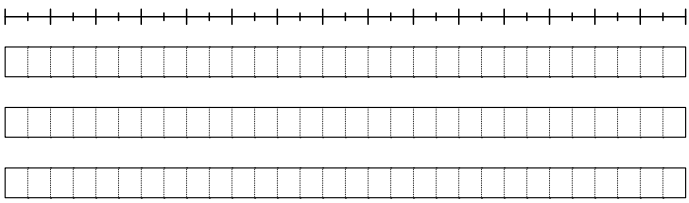
ア　現在処理をしている割込み処理よりも、優先順位の高い割込みをマスクをする。
イ　現在処理をしている割込み処理よりも、優先順位の低い割込みをマスクをする。
ウ　現在処理をしている割込み処理を中断して、後から発生した割込み処理を優先する。
エ　現在処理をしている割込み処理を優先して、後から発生した割込み処理を待たせる。

□第１種　平成 12 年度（2000 年度）午前・問 21

表に示す３種類の割込み種別をもち、多重割込みを許すリアルタイムシステムがある。割込みＣが発生した 20 ミリ秒後に割込みＡが発生し、割込みＡ発生後 80 ミリ秒後に割込みＢが発生した。この場合, 割込みＣの処理が終了するまでの時間は何ミリ秒か。ここで、割込みの優先度はＡ＞Ｂ＞Ｃとし、OS などのオーバヘッドは無視するものとする。

割込み種別	周期（ミリ秒）	割込み処理時間（ミリ秒）
A	50	10
B	非同期	30
C	非同期	85

ア　105　　イ　135　　ウ　145　　エ　155



CPUスケジューリング

CPU は、プロセスの間で切り替えられることは既に学習しました。切り替えの方式てであるスケジューリングには、多種多様な方法が存在します。

到着順　FIFO/FCFS（First-In First-Out / First-come First-served）
処理時間順　SPT（shortest-processing time-first）
優先度順　（priority scheduling）
ラウンドロビン　RR（round robin）
多重待ち行列　（multi-level queue）
　⋮

性能評価基準
(a) 応答時間（turnaround time ／ response time）

(b) スループット（throughtput）

・予測性

・公平性

・グレースフルな終了

■第２種　平成 12 年度（2000 年度）秋期 午前・問 30

システムが単位時間内にジョブを処理する能力の評価尺度はどれか。

ア　MIPS 値　　イ　応答時間
ウ　スループット　　エ　ターンアラウンドタイム

到着順　FIFO（Fisrt–In Fisrt–Out）／FCFS（First–come First–served）

ジョブを到着順に実行する。ジョブはいったん開始されると完了まで実行されます。すなわち、ノンプリエンプティブです。

ジョブの到着

○ … ○ ○ ○　　○ CPU

公平かつ単純であるが、あまり優れた方法であるとは言えない。
応答時間が長くなるため TSS には適さない。
本来優先すべきタスクであっても、先に実行されるとは限らない。

フォーク並び

処理時間順　SPT（shortest-processing time-first）／SJF（shortest job first）

処理時間の短いジョブから実行します。平均応答時間を最小にし、利用効率を向上させやすいという特徴があります。ノンプリエンプティブな方式です。

ジョブの到着

○ … ○ ○ ○　　○ CPU

予め処理時間を知ることは困難である。

プリエンプションと組み合わせると効果的。

実行を一時中断し、他のジョブを実行する。

優先度順 （priority scheduling）

各プロセスに優先度を与え、優先度の順に実行する方式です。

ジョブの到着

○ … ○ ○ ○　○ CPU

より優先度の高いジョブが到着するか、
待ち状態のジョブがより高い優先度を得ると、
現在のジョブは中断される。

◆優先順位：

1. リアルタイムプロセス
2. オペレーティングシステム
3. 対話型ジョブ
4. バッチジョブ（バックグラウンドジョブ）

■静的優先度方式

最初に与えられた優先度は変化しない。

■動的優先度方式

優先度が状況に応じて動的に変化する。

優先度の低いプロセスが待たされる**無期延期＝スタベーション**（*starvation*）が発生しやすい

対策：**エージング**（*aging*）

ラウンドロビン　RR（round robin）

各プロセスに、タイムクウォンタムを割り当てて、そのプロセスがその時間内に終了しないと、実行可能行列の末尾に移されます。対話型ジョブの応答時間を短くします。

ジョブの到着

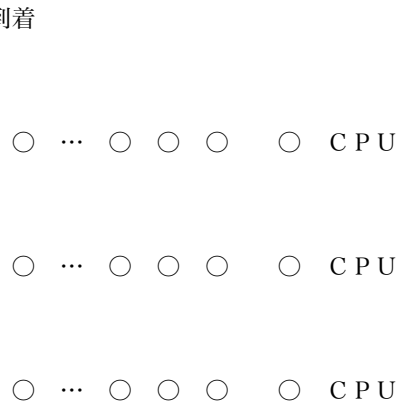
○ … ○ ○ ○　　○ ＣＰＵ

タイムクウォンタムを消費すると、現在のジョブは中断されて、
待ち行列の末尾に戻される。

タイムクウォンタムを無限大にするとFCFSと同じ。

多重待ち行列

プロセスの特性などから、複数の待ち行列となるようなグループに分け、各グループを割り当てられた規則で切り替えながら処理していく方式です。



各待ち行列内のスケジューリングに加えて、キュー間のスケジューリングが必要となります。そのスケジューリングにも、優先度順やラウンドロビンなどを適用します。

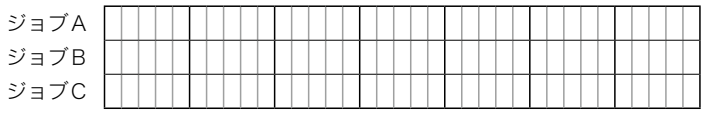
多重待ち行列方式では、プロセスは、一つの待ち行列に静的に割り当てられています。これに対し、待ち行列間での動的な移動を伴うのが、**フィードバック待ち行列**方式です。たとえば、最初に高い優先順位と短いCPU時間を割り当て、その後は優先度を低くし、CPU時間を徐々に長くします。

応答時間の評価

三つのジョブが、以下のように与えられています。応答時間を求めて、スケジューリング方式による違いを検討します。

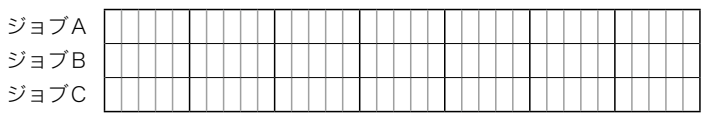
	到着時刻	処理時間
ジョブA	Ø	1Ø
ジョブB	2	2Ø
ジョブC	6	5

①到着順（FIFO ／ FCFS）



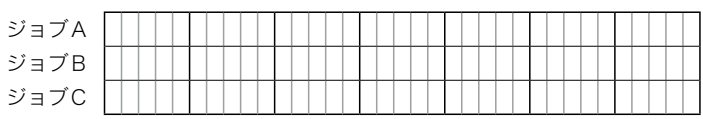
完了時刻	応答時間

②ラウンドロビン（RR）：タイムクウォンタム＝1とする



完了時刻	応答時間

③処理時間順（SPT ／ SJF）



完了時刻	応答時間

処理時間・利用効率の評価

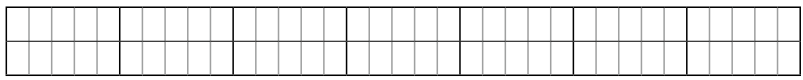
以下のスケジューリングを考え、処理時間と利用効率を計算します。

・処理開始時直前に、ジョブＡ→ジョブＢ→ジョブＣの順にジョブが投入されています。
・各処理において、ＣＰＵ処理が完了した後に、入出力処理が行われます。
・三つのジョブを一連の作業と考えると、この作業は２回行われます。

	CPU 処理	入出力処理
ジョブA	35	10
ジョブB	20	20
ジョブC	5	25

①到着順

CPU
I/O



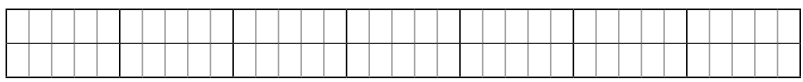
CPU の利用効率 ＝

I/O の利用効率 ＝

平均応答時間 ＝

②処理時間順（SPT ／ SJF）

CPU
I/O



CPU の利用効率 ＝

I/O の利用効率 ＝

平均応答時間 ＝

MS–Windows における優先度

MS-Windows では、プロセスの優先度を自由に変更できるようになっています。以下のように行います。

① [Shift] + [Ctrl] + [ESC] でタスクマネージャを起動する。

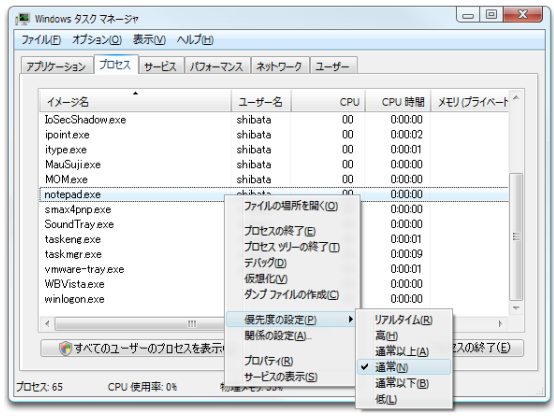
②“プロセス”のタブを選択する。

③変更したいプロセスで右ボタンをクリックし、“優先度の設定”をクリックします。

④優先度を選択します。

▶ 内部的な優先度は、0～31 の 32 段階ですが、ユーザが指定できるのは 6 段階です。

指定する優先度	基本優先度
リアルタイム	24
高	13
通常以上	10
通常	8
通常以下	6
低	4



一般的にアプリケーションを起動すると、“通常”の優先度とみなされ、優先度は 8 となります（多少上下することもあります）。

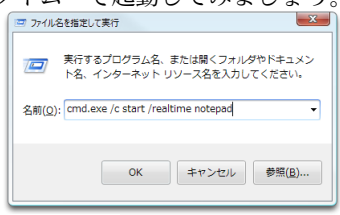
start コマンドを利用すると、アプリケーションの起動時に優先度を指定できます。

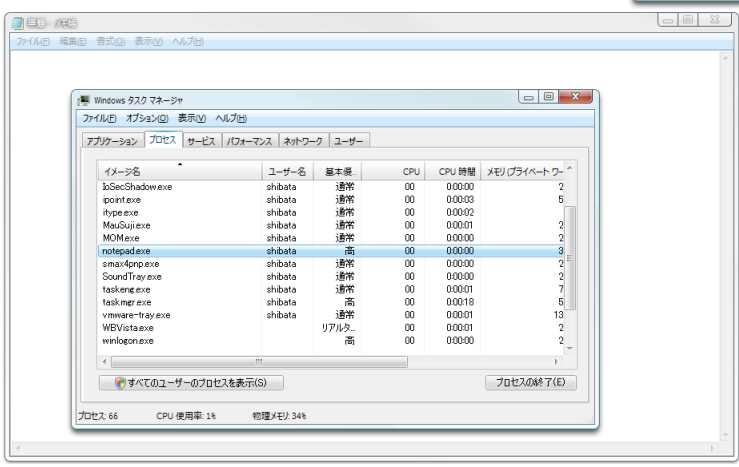
ここでは、メモ帳（notepad.exe）を、優先度“リアルタイム”で起動してみましょう。

① [⊞] + R で、『ファイル名を指定して実行』ダイアログを表示させます。

②以下のように打ち込みます。

```
cmd.exe /c start /realtime notepad
```





指定する優先度
realtime
high
abovenormal
normal
belownormal
low

□ソフトウェア　平成 13 年度（2001 年度）午前・問 26

リアルタイム OS のプリエンプションの記述として、適切なものはどれか。

ア　一定時間ごとに実行中のタスクを中断して、別の実行可能なタスクを実行する。
イ　実行中のタスクが終了または待ち状態になった場合、別の実行可能なタスクを実行する。
ウ　実行中のタスクよりも優先順位の高いタスクが実行可能状態になった場合、実行中のタスクを中断してその優先順位の高いタスクを実行する。
エ　優先順位の高いタスクが優先順位の低いタスクの終了を待っている場合、優先順位の低いタスクの優先順位を一時的に高くして実行する。

□第１種　平成 11 年度（1999 年度）午前・問 23

ノンプリエンプティブなマルチタスク OS の欠点はどれか。

ア　アプリケーションプログラムがループすると OS に制御が戻らない。
イ　仮想記憶のページ置換えの機能が不十分となる。
ウ　カーネルモードとユーザモードの区分がなく、OS が破壊される危険性がある。
エ　割込み処理が遅れ、入出力処理のリアルタイム性が劣る。

□第１種　平成 11 年度（1999 年度）午前・問 99

リアルタイム OS におけるタスクの優先順位に関する記述として、正しいものはどれか。

ア　I/O の完了やタイマの完了、タスク間の連絡など異なったイベントで待ち状態になったタスクが複数ある場合、タスクは優先順位に従って実行可能状態になる。
イ　システムのデバイスドライバから I/O の完了を通知されたタスクは、優先順位に関係なく一定時間実行状態になる。
ウ　セマフォを使用して共有資源を占有していても、優先順位の高いタスクが同じ資源を要求すると、優先順位の低いタスクは資源の占有を解かれてしまう。
エ　優先順位の低いタスクは、長い時間実行可能状態にあっても、優先順位のより高いタスクが実行状態にある間は実行状態になれない。

□第１種　平成 11 年度（1999 年度）午前・問 98

タスクが実行状態（RUN）、実行可能状態（READY）、待ち状態（WAIT）の三つの状態で管理されるリアルタイムＯＳにおいて、タスクＡ，Ｂ，Ｃがプリエンプティブなスケジューリングによって、図に示すとおりに状態遷移した。各タスクの優先順位の関係のうち、正しいものはどれか。ここで、優先順位の関係は“高い＞低い”で表す。

タスクＡ	RUN	WAIT		READY	RUN	READY
タスクＢ	WAIT	RUN	WAIT	RUN	WAIT	
タスクＣ	WAIT	READY	RUN	WAIT		RUN

ア　Ａ＞Ｂ＞Ｃ　　イ　Ｂ＞Ａ＞Ｃ　　ウ　Ｂ＞Ｃ＞Ａ　　エ　Ｃ＞Ｂ＞Ａ

□第１種　平成 10 年度（1998 年度）午前・問 23

プロセススケジューリングに関する記述のうち、ラウンドロビン方式の説明として正しいものはどれか。

ア　各プロセスに優先順位が付けられる。後に到着したプロセスの優先順位が処理中のプロセスよりも高ければ、処理中のものを中断し、到着プロセスを処理する。
イ　各プロセスの処理時間に比例して、タイムクウォンタム（CPU 割当て時間）を変更する。
ウ　各プロセスを待ち行列に置かれた順にタイムクウォンタムずつ処理し、終了しないときは同じ優先順位の待ち行列の最後尾につなぐ。
エ　入出力の終了やコマンド入力などの割込みを引き金にして、次に実行するプロセスを決める。

□第１種　平成７年度（1995 年度）午前・問 26

オペレーティングシステムにおけるタスクのスケジューリングに関して、正しいものを二つ選べ。

ア　FIFO（First–In First–Out）方式では、要求された順番に CPU 時間を割り当てるので、処理時間の短いタスクほど順番が有利となる。
イ　RR（Round Robin）方式は、要求された順番に CPU 時間を割り当て、割り当てられた時間を使い切ったあとは、待ち行列の最後に回す方式である。
ウ　SJF（Shortest Job First）方式は、JCL などで指定した処理時間の短いタスクを先に処理する方式で、処理時間の長いタスクほど後回しになる。
エ　多重待ち行列方式は、最初に低い優先順位と長い CPU 時間を割り当て、その後は優先度を高くし、CPU 時間を徐々に短くする方式である。
オ　優先度順（Priority）方式は、各タスクに優先順位を与えて、順位の高い順に CPU 時間を割り当てるので、優先度の低いタスクでも CPU 使用率が高いと優先度が上がり、不利にならない方式である。

□ソフトウェア　平成 13 年度（2001 年度）午前・問 25

OS におけるタスクのスケジューリングに関する記述として、適切なものはどれか。

ア　多重待ち行列方式は、割り当て要求のあったタスクに対して最初に低い優先順位と長い CPU 時間を割り当て、その後は優先度を高くし、CPU 時間を徐々に短くする方式である。
イ　到着順方式では、タスクが生成された順に高い優先順位を付けて CPU 時間を割り当てる。これは先に開始されたタスクを優先させて、早く終了させることを目的としているからである。
ウ　優先順位方式では、CPU の利用状況の低いタスクの優先順位を順次高くし、逆に CPU を多く利用したタスクの優先順位を低くするので、システム全体の処理効率を高めるのに適している。
エ　ラウンドロビン方式は、要求された順番に CPU 時間を割り当て、割り当てられた時間を使い切った後は、待ち行列の末尾に回す方式である。

□第１種　平成 12 年度（2000 年度）午前・問 25

プロセスのスケジューリング方式におけるフィードバック待ち行列方式に関する記述として、適切なものはどれか。

ア　一定時間内に処理が終了しない場合は、順次優先度を落としていく方式である。
イ　実行待ちリストで待っているプロセスの中で、推定実行時間が最も短いものを次に選ぶ方式である。
ウ　実行待ちリストに到着した順に従って、プロセスを実行する方式である。
エ　プロセスを到着順に実行し、一定時間内に終了しない場合は、実行待ちリストの最後尾につなぐ方式である。

■第２種　平成 11 年度（1999 年度）春期 午前・問 13

Ａ及びＢの二つのプログラムがあり、それぞれ単体で実行したときの処理装置（CPU）、入出力装置（I/O）の占有時間は、図のとおりである。プログラムＡ、Ｂを１台の CPU のもとで起動したとき、プログラムＢが最短で終了するのは起動の何ミリ秒後となるか。ここで、プログラム等の実行条件は次のとおりとする。

① プログラムの実行優先度はＡのほうがＢより高い。
② プログラムＡ、Ｂは同一の入出力装置を利用する。
③ CPU 処理を実行中のプログラムは、入出力装置を行うまでは実行を中断されない。
④ 入出力装置も入出力処理が終了するまで実行を中断されない。
⑤ CPU 処理の切替え（タスクスイッチ）に必要な時間は無視できる。

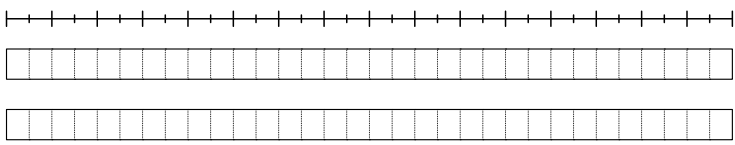
プログラムＡ　　ミリ秒

CPU	→	I/O	→	CPU	→	I/O	→	CPU
2Ø		3Ø		2Ø		4Ø		1Ø

プログラムＢ　　ミリ秒

CPU	→	I/O	→	CPU	→	I/O	→	CPU
1Ø		3Ø		2Ø		2Ø		2Ø

ア　100　　イ　120　　ウ　140　　エ　160　　オ　180

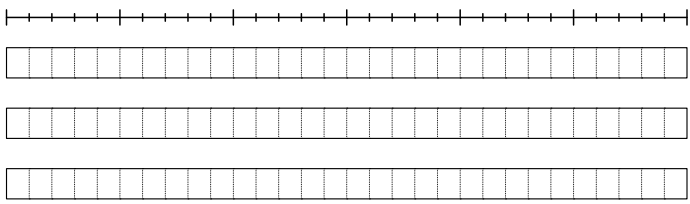


■第２種　平成 12 年度（2000 年度）春期 午前・問 35

三つのタスクの優先度及び各タスクを単体で実行した場合の処理装置（CPU）、入出力装置（I/O）の占有時間は、表のとおりである。三つのタスクが同時に実行可能状態になってから、すべてが終了するまでの CPU のアイドル時間は何ミリ秒か。ここで、CPU は 1 個とし、各タスクの I/O 処理は並行して処理可能であり、OS のオーバヘッドは無視できるものとする。

タスク	優先度	単独動作時の所要時間
A	高	各タスクともに、 CPU 5 ミリ秒 → I/O 8 ミリ秒→ CPU 2 ミリ秒
B	中	
C	低	

ア　3　　イ　4　　ウ　5　　エ　6

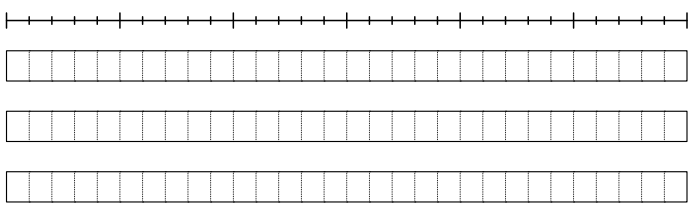


■第２種　平成 12 年度（2000 年度）秋期 午前・問 32

三つのタスクの優先度、各タスクを単体で実行した場合の処理装置（CPU）と入出力装置（I/O）の占有時間は、表のとおりである。優先順位方式のタスクスケジューリングを行う OS のもとで、三つのタスクが同時に実行可能状態になってから、タスク C が終了するまでの間に、タスク C が実行可能状態にある時間は延べ何ミリ秒か。ここで、各タスクの入出力は並行して処理が可能であり、OS のオーバヘッドは無視できるものとする。

タスク	優先度	単独実行の占有時間（単位：ミリ秒）				
		CPU → I/O → CPU → I/O → CPU				
A	高	4	4	3	5	3
B	中	2	6	3	6	2
C	低	2	5	3	4	1

ア　2　　イ　5　　ウ　8　　エ　11



□第１種　平成８年度（1996 年度）午前・問 29

プロセスＡ，Ｂはそれぞれ次のような時間（ミリ秒）で CPU と I/O（入出力装置）を用いた処理をする。

プロセスＡ：CPU(10) → I/O(20) → CPU(10) → I/O(30) → CPU(10)

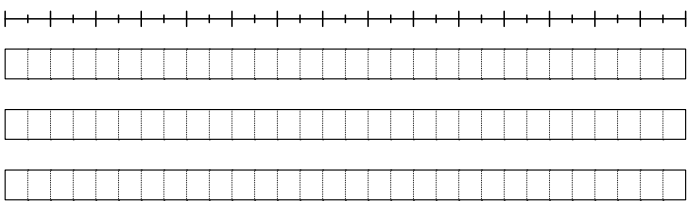
プロセスＢ：CPU(5) → I/O(30) → CPU(5) → I/O(20) → CPU(5)

（→はプロセスＡ，Ｂの進行過程を示し、( ) 内の数字は処理時間を示す。）

この二つのプロセスを同時に実行させたとき、プロセスの開始から二つのプロセスの完了まで間の CPU 利用率（%）に最も近いものはどれか。ここで、同期実行の条件は次のとおりとする。

・最初はプロセスＡに CPU 時間が割り当てられる。
・CPU 時間は５ミリ秒ごとのラウンドロビン方式でスケジューリングされる。
・二つのプロセスがアクセスする入出力装置は異なった装置であり、同時処理が可能である。
・割込み処理に要する時間は無視してよい。
・この二つのプロセスだけが実行され、ほかにプロセスはない。

ア　40　　イ　45　　ウ　50　　エ　55　　オ　60



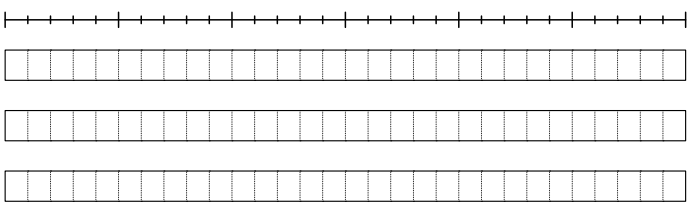
■基本　平成 20 年度（2008 年度）秋期　午前・問 32

処理はすべて CPU 処理である三つのジョブＡ、Ｂ、Ｃがある。それらを単独で実行したときの処理時間は、ジョブＡが５分、ジョブＢが 10 分、ジョブＣは 15 分である。この三つのジョブを次のスケジューリング方式に基づいて同時に実行すると、ジョブＢが終了するまでの経過時間はおよそ何分か。

〔スケジューリング方式〕

(1)　一定時間（これをタイムクウォンタムと呼ぶ）内に処理が終了しなければ、処理を中断させて、待ち行列の最後尾へ回す。
(2)　待ち行列に並んだ順に実行する。
(3)　タイムクウォンタムは、ジョブの処理時間に比べて十分に小さい値とする。
(4)　ジョブの切替え時間は考慮しないものとする。

ア　15　　イ　20　　ウ　25　　エ　30



プロセスとスレッド

オペレーティングシステムから見たプログラムの実行単位が**プロセス**と**スレッド**です。

▪プロセス

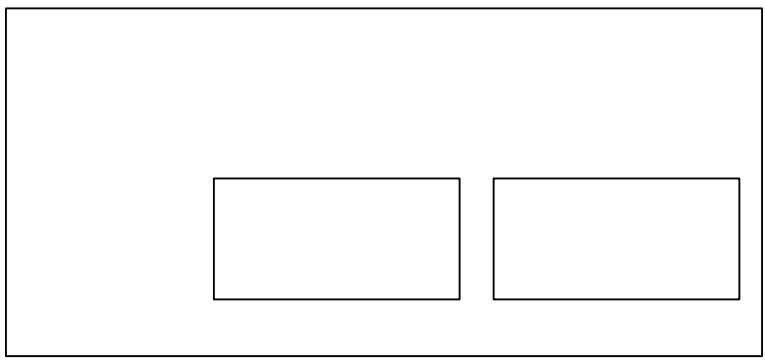
オペレーティングシステムから見たプログラムの実行単位である**プロセス**（*process*）は、実行中のプログラムのアドレス空間など、実行環境を含むものです。アドレス空間を分割して実行できるため、並行処理であるパイプ処理やバックグラウンド処理にも適しています。しかし、以下のような欠点があります。

・各プロセスが別個のアドレス空間を利用するために多くの資源が必要である
・生成や切替えに時間がかかる

そのため**重量プロセス**（ヘビーウェイトプロセス）とも呼ばれます。

▪スレッド

上記のようなプロセスの問題を解決するのが、**軽量プロセス**（ライトウェイトプロセス）とも呼ばれる**スレッド**（*thread*）です。スレッドは、プロセスを細分化した基本的かつ最小の処理単位です。高速で生成できる、スワップの負荷が小さいなどの特徴があります。**CPU 資源のみが割り当てられて、スタック、プログラムカウンタ、レジスタセットを保持しますが、その他の資源は親のプロセスから継承します**。同一のプロセスから発生するスレッドは、同じアドレス空間を継承し共有しなければなりません。分散処理や、マルチ CPU での並行処理などで利用されます。



【演習】Windows の『タスクマネージャー』で、プロセスとスレッドを見てみよう！！

□第１種　平成８年度（1996 年度）午前・問 26

プログラムの実行に必要なシステム資源の割当ての単位であるプロセスとは異なり、軽量プロセスとも呼ばれ、CPU 以外の資源は割り当てられず、親のプロセスから必要な資源を継承するものはどれか。

ア　カーネル　　イ　コンテキスト　　ウ　ジョブステップ
エ　スレッド　　オ　タスク

□第１種　平成 10 年度（1998 年度）午前・問 25

並行処理の単位として、プロセスのほかに、プロセス内に複数存在するスレッドを用いることがある。一つのプロセス内のスレッドが共有するものはどれか。

ア　アドレス空間　　イ　スタック
ウ　プログラムカウンタの値　　エ　レジスタセットの値

▪セマフォ

セマフォ（*semaphore*）とは、手旗信号や腕木式信号機を意味する言葉です。コンピュータの世界では、E.W.Dijkstar が提案した、カウンタと待ち行列からなるデータ構造を指し、プロセスの排他制御のために用いられます。

資源の状態を待ち（P ＝ passeren 資源獲得）、続行（V ＝ verhoog 資源解放）の二つの操作で表すことで、共有データの排他的アクセス、同一資源の割当てなどの作業を行い、プロセス間で同期を取って、資源を共有する複数のプロセスの処理を円滑に行います。

クリティカルセクション（各プログラムが共有資源へのアクセスを行う局面）開始前に行うのがP操作で、終了時に行うのがV操作です。待ち状態のプロセスがある場合にV操作を行うと、そのプロセスの一つが実行可能状態に移ることになります。

▶　腕木式信号機とは、鉄道の単線区間で列車の衝突を防ぐために、同時には一つの列車しかその区間に入れないようにするための信号機のことです。初期の腕木式信号機が示す合図は、「Passeren 進入可能」と「Verhoog 進入不可」の２種類でした。

腕木式信号機のように、0 と 1 のみの値をとるセマフォを２値セマフォ（binary semaphore）と言います。値が任意の個数（たとえば 0 から n−1 までの n 個を表す）を表すセマフォもあり、そのようなセマフォは計数セマフォ（counting semaphore）と呼ばれます。

二つのプロセスA,Bが、共用する変数xに1を加算して、xを0から2にするという例を考えます。

［例1］

・A … xの値0を読み込む。
・A … 1を加算した1をxに書き込む。
・B … xの値1を読み込む。
・B … 1を加算した2をxに書き込む。

これで、xは2となります。もっとも、同期をとって処理すべき変数xを、非同期で処理してしまうと、次のようになる可能性もあります。

［例2］

・A … xの値0を読み込む。
・B … xの値0を読み込む。
・A … 1を加算した1をxに書き込む。
・B … 1を加算した1をxに書き込む。

これだと、xは1となってしまいます。

上の問題を解決するために、ロック用の変数yを導入します。変数yが0であれば別のプロセスはxに対して処理をせず、1なら処理をすることにします。すなわち各プロセスは、xに対して処理する前にyをチェック・変更します。［例2］は以下のようになります。

・A … yの値を読み込む。
・A … yが1なので0に変更する。
・A … xの値0を読み込む。
・B … yの値を読み込む。
・B … yが0なので待つ。
・A … 1を加算した1をxに書き込む。
・A … yの値を1に戻す。
・B … yの値を読み込む。
・B … yが1なので0に変更する。
・B … xの値0を読み込む。
・B … 1を加算した1をxに書き込む。
・B … yの値を1に戻す。

これで、xは2となります。これでうまくいくように見えるが、yに対しても最初のxと同じ状況が発生することになります。実際には、あるプロセスがセマフォを読み込んで値を変更するまでの間、他のプロセスはセマフォに対して処理しないことが保証されています。

List 1　ThreadPrinter.java

```
// ThreadPrinterクラス  Copyrigth 2009 by BohYoh Shibata

import java.util.*;

/**
 *  クラスThreadPrinterは、スレッドごとにタブを与えて標準出力ストリームに
 *  表示するためのクラスです。
 *
 *  <p>呼び出せるメソッドはprintlnのみです。
 *  あるスレッドからThreadPrinter.println("A")と呼び出され、その1.5秒後に
 *  別のスレッドからThreadPrinter.println("B")と呼び出された場合は、
 *  以下のように表示されます。
 *  <blockquote><pre>
 *  000.000 A
 *  001.500         B
 *  </pre></blockquote>
 *  タブ幅は実行環境に依存します。
 *
 * @author  柴田望洋
 * @version 1.0 23/05/2009
 *
 */
public class ThreadPrinter {
    private static long start = System.nanoTime();
    private static Map<Thread, Integer> tabMap = new HashMap<Thread, Integer>();

    // 現在実行中のスレッドのタブを取得・設定
    private static int getTab() {
        Thread thread = Thread.currentThread();
        if (tabMap.containsKey(thread)) {   // 登録済み
            return tabMap.get(thread);
        } else {                            // 未登録
            int tab = tabMap.size();
            tabMap.put(thread, tab);
            return tab;
        }
    }

    /**
     *  文字列sを表示します。
     *  各行の先頭には、本メソッドを初めて呼び出してからの経過時間を
     *  ミリ秒単位で"###.###"形式で出力します。
     * @param   s 出力される文字列。
     */
    public synchronized static void println(String s) {
        long t = (System.nanoTime() - start) / (1000 * 1000);
        System.out.print(String.format("%03d.%03d", t / 1000, t % 1000));
        for (int i = 0; i <= getTab(); i++)
            System.out.print("\t");
        System.out.println(s);
    }
}
```

List 2 Sleep.java

```
// sleepクラス  Copyrigth 2009 by BohYoh Shibata

/**
 * クラスSleepは、システムタイマーとスケジューラが正確であることを前提として、
 * 現在実行中のスレッドをスリープさせるためのユーティリティクラスです。
 *
 * @author  柴田望洋
 * @version 1.0 23/05/2009
 */
public class Sleep {
    /**
     * システムタイマーとスケジューラが正確であることを前提として、現在実行中の
     * スレッドを、指定されたミリ秒数の間、スリープ（一時的に実行を停止）
     * させます。スレッドはモニターの所有権を失いません。
     * InterruptedException例外が発生した場合は、スタックトレースを
     * 標準エラーストリームに出力します。その際、現在のスレッドの
     * 「割り込みステータス」はクリアされます。
     *
     * @param   millis ミリ秒単位のスリープ時間の長さ。
     *          millisミリ秒スリープします。
     * @see     #sleepLt()
     * @see     Thread#sleep()
     */
    public static void sleep(long mills) {
        try {
            Thread.sleep(mills);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    /**
     * システムタイマーとスケジューラが正確であることを前提として、現在実行中の
     * スレッドを、指定されたミリ秒数より短い間、スリープ（一時的に実行を停止）
     * させます。スレッドはモニターの所有権を失いません。
     * InterruptedException例外が発生した場合は、スタックトレースを
     * 標準エラーストリームに出力します。その際、現在のスレッドの
     * 「割り込みステータス」はクリアされます。
     *
     * @param   millis ミリ秒単位のスリープ時間の長さ。
     *          0以上millis未満スリープします。
     * @see     #sleep()
     * @see     Thread#sleep()
     */
    public static void sleepLt(long mills) {
        sleep((long)(Math.random() * mills));
    }
}
```

List 3　MyThreadTester.java

```
// 二つのスレッド（各スレッドは変数xの値を取り出して1を加える）
class MyThread implements Runnable {
    static int x;

    public synchronized void go() {
        int temp = x;                // xの値を取り出す
        ThreadPrinter.println("→" + temp);
        Sleep.sleep(200);            // 200ミリ秒スリープ
        x = temp + 1;                // 取り出した値に1を加えてxに代入する
        ThreadPrinter.println("←" + x);
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            go();
            Sleep.sleep(200);        // 200ミリ秒スリープ
        }
    }
}

public class MyThreadTester {
    public static void main(String[] args) {
        MyThread job = new MyThread();
        Thread a = new Thread(job);      // スレッドaを作成
        Thread b = new Thread(job);      // スレッドbを作成
        a.start();                       // スレッドaの実行開始
        b.start();                       // スレッドbの実行開始
    }
}
```

プログラムの実行結果。
各スレッドのgoメソッドが一気に実行されるため、変数xの最終的な値は必ず10になる。

```
000.000  →0
000.232  ←1
000.232           →1
000.433           ←2
000.434  →2
000.636  ←3
000.638           →3
000.839           ←4
000.840  →4
001.040  ←5
001.041           →5
001.241           ←6
001.242  →6
001.443  ←7
001.444           →7
001.644           ←8
001.645  →8
001.848  ←9
001.849           →9
002.050           ←10
```

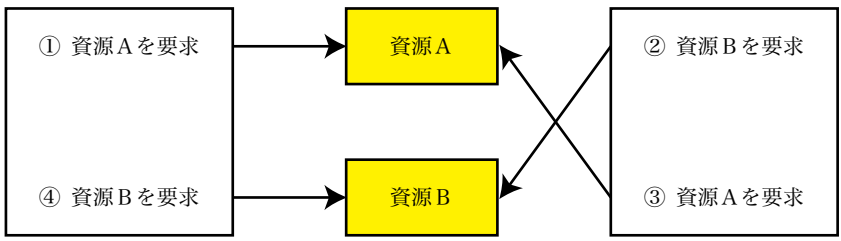
synchronizedが省略された場合の実行結果。
各スレッドのgoメソッドが一気に実行される保証がないため、変数xの最終的な値が10になるとは限らない。

```
000.000  →0
000.034           →0
000.234           ←1
000.234  ←1
000.434           →1
000.435  →1
000.635           ←2
000.638  ←2
000.849           →2
000.849  →2
001.079  ←3
001.124           ←3
001.353           →3
001.354  →3
001.627  ←4
001.628           ←4
001.828  →4
001.829           →4
002.029  ←5
002.031           ←5
```

▪デッドロック

デッドロック（*deadlock*）とは、複数のプロセスが互いに次にアクセスする資源にロックをかけてしまい、永久に待ちの状態に陥ることです。

例えば、プロセスXが共有資源Aを、プロセスYが共有資源Bを排他的に使用しているとき、プロセスXは共有資源Bの解放を、プロセスYは共有資源Aの解放を待っている場合では、プロセスX、Yともに必要な資源を利用することができません。デッドロックが発生した結果、プロセスの続行ができなくなります。



デッドロックを回避するためには、両方のプロセスでの資源の獲得順序を同一にします。

■第2種　平成9年度（1997年度）春期 午前・問33

プロセスの相互排除（排他制御）に用いられるものはどれか。

ア　コンテンション　　イ　セマフォ
ウ　チェックポイント　　エ　ハッシュ

■第2種　平成12年度（2000年度）秋期 午前・問29

プロセスの相互排除（排他制御）に用いられるものはどれか。

ア　コンテンション　　イ　セマフォ
ウ　チェックポイント　　エ　ハッシュ

□第１種　平成９年度（1997 年度）午前・問 27

セマフォを用いる目的として、正しいものはどれか。

ア　共有資源を管理する。
イ　スタックを容易に実現する。
ウ　スラッシングを減らす。
エ　セグメンテーションを実現する。

□第１種　平成７年度（1995 年度）午前・問 28

オペレーティングシステムの機能である同期制御に関する記述として、正しいものはどれか。

ア　Post / Wait マクロ命令は、リソースの占有使用要求及び解放に用いる。
イ　セマフォは、タスクの同期を取るために用いる。
ウ　ダブルバッファリング方式は、タスクの同期制御を実現する一つの手段である。
エ　データ指向モデルは、並行処理の動作を簡潔に表現できるモデルである。

□第１種　平成 11 年度（1999 年度）午前・問 21

次のケースのうち、デッドロックが起こるものはどれか。

ア　タスクＸが資源Ａと資源Ｂを確保した後に、タスクＹが資源Ａと資源Ｂを共有した。
イ　タスクＸが資源Ａを確保し、タスクＹが資源Ｂを確保した後に、タスクＹが資源Ａを要求し、タスクＸが資源Ｂを要求した。
ウ　タスクＸが資源Ａを確保し、タスクＹが資源Ｂを確保した後に、タスクＹが資源Ｃを要求し、タスクＸが資源Ｂを要求した。
エ　タスクＸが資源Ａを確保し、タスクＹが資源Ｂを確保した後に、タスクＺが資源Ａと資源Ｂを要求した。

□ソフト　平成 17 年度（2005 年度）終期 午前・問 25

資源Ｘ，Ｙ，Ｚを占有して処理を行うプロセスＡ～Ｄがある。各プロセスは処理の進行に伴い、表中の数値の順に資源を占有し、実行終了後に三つの資源を一括して解放する。プロセスＡとデッドロックの関係を起こす可能性のあるプロセスはどれか。

プロセス名	資源の占有順序		
	資源Ｘ	資源Ｙ	資源Ｚ
プロセスＡ	1	2	3
プロセスＢ	3	2	1
プロセスＣ	2	3	1
プロセスＤ	1	2	3

ア　Ｂ、Ｃ
イ　Ｂ、Ｃ、Ｄ
ウ　Ｂだけ
エ　Ｃだけ

□ソフト　平成 14 年度（2002 年度）午前・問 28　（ほぼ同じ）
□第１種　平成９年度（1997 年度）午前・問 28　（ほぼ同じ）

ファイル管理

データの長期保存は、OSの重要な役割の一つであり、**ファイル**（*file*）と呼ばれる形式で保存を行います。

ディレクトリ

膨大な数のファイルを1元的に管理するのは困難であるため、UNIXやMS-Windowsなどの OSでは、**ディレクトリ**（directory）の概念が導入されています。

ディレクトリは、複数のファイルをまとめてグループ化するためのものであると考えましょう。すなわち、ディレクトリには、任意の個数（0個以上）のファイルを格納することができます。

また、ディレクトリは、階層的な構造をもつことができます。

▶ Macや最近のWindowsでは、ディレクトリでなく“フォルダ”と呼ばれます。もともと、directoryは『住所録』『名鑑』、folderは『紙ばさみ』『書類ばさみ』という意味の語句です。

▪ ルートディレクトリ（root directory）

最も上位に位置するディレクトリであり、1個のみが存在します。

UNIXでは/と表し、日本語版Windowsでは¥と表します。

▪ サブディレクトリ（sub directory）

階層的なディレクトリ構造において、下位に位置するディレクトリをサブディレクトリあるいは子ディレクトリと呼びます。

▪ 親ディレクトリ（super directory）

階層的なディレクトリ構造において、上位に位置するディレクトリをスーパーディレクトリあるいは親ディレクトリと呼びます。ルートディレクトリ以外のディレクトリは、必ず1個の親ディレクトリをもちます。

親ディレクトリを表す記号は“..”です。

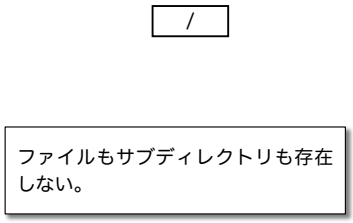
▶ 自分自身のディレクトリを表す記号は、“.”です。

※ルートディレクトリを含め、各ディレクトリは、任意の個数（0個以上）のファイルおよび任意の個数のサブディレクトリをもつことができます。

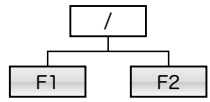
【演習】Windowsの『エクスプローラ』で、ドライブCのディレクトリとファイルを見てみよう！！

ディレクトリとファイルの例

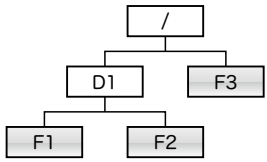
ディレクトリとファイルの構成例を示します。



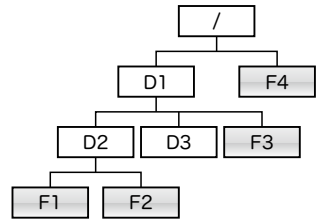
ファイルもサブディレクトリも存在しない。



ルートディレクトリの下にファイルF1およびF2が存在。サブディレクトリは存在しない。



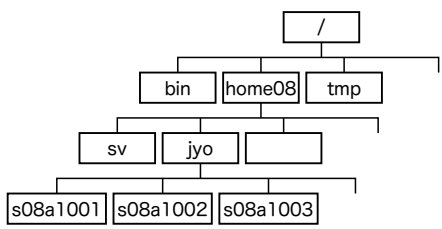
ルートディレクトリの下にディレクトリD1およびファイルF3が存在。サブディレクトリD1の下には、ファイルF1およびF2が存在。
※ディレクトリD1の親ディレクトリは、ルートディレクトリ



ディレクトリD3の下は空。
※ディレクトリD2およびD3の親ディレクトリは、いずれもD1。

本学でのディレクトリ構成

本学センターでは、ルートディレクトリの下に、多くのサブディレクトリが存在します。2008 年 4 月入学の学生用のサブディレクトリが、home08 であり、その下に、各学科用のサブディレクトリが用意されています。情報工学科用のディレクトリは jyo であり、その下に各個人用のディレクトリ s08a1001, s08a1002, s08a1003, … が存在します。



【演習】UNIX（Linux）で、センターのディレクトリを見てみよう！！

パス

任意のファイルやディレクトリを表す方式として、以下に示す二つがあります。

- 絶対パス

ルートディレクトリからの全ての経路で表します。

- 相対パス

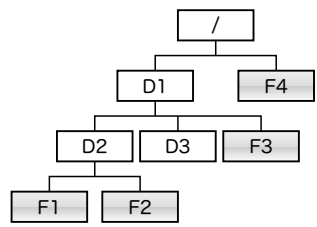
現在作業を行っている**カレントディレクトリ**（*current directory*）／**ワーキングディレクトリ**（*working directory*）からの経路で表します。

パスの表現では、以下の記号を利用します。

..	親ディレクトリを表します。
.	そのディレクトリ自身を表します。
~	ホームディレクトリを表します。
/	パス表現の先頭にある場合はルートディレクトリを表し、 中間にある場合は、ディレクトリ名又はファイル名の区切りを表します。 ※日本語版 Windows では￥を用います。

※絶対パスは、ルートディレクトリからの経路で表すため、カレントディレクトリとは無関係に決定します。
※相対パスは、カレントディレクトリからの経路で表すため、カレントディレクトリに依存します。

左図において、各ファイルの絶対パス、相対パスを考えてみましょう。



絶対パス

ファイル F1	/D1/D2/F1
ファイル F3	/D1/F3
ファイル F4	/F4
ディレクトリ D3	/D1/D3

相対パス

カレントディレクトリが D2 であるとき

ファイル F1	F1
ファイル F3	../F3
ファイル F4	../../F4
ディレクトリ D3	../D3

カレントディレクトリが D1 であるとき

ファイル F1	D2/F1
ファイル F3	F3
ファイル F4	../F4
ディレクトリ D3	D3

■第２種　平成８年度（1996 年度）秋期 午前・問 31

階層構造のディレクトリを用いたファイル管理を行うオペレーティングシステムにおいて、ファイルが置かれているディレクトリを指すときに指定するものはどれか。

ア　拡張子　　イ　サブディレクトリ　　ウ　パス
エ　ルートディレクトリ　　オ　ワイルドカード

■第２種　平成７年度（1995 年度）秋期 午前・問 49

MS–DOS や UNIX におけるディレクトリ構造に関する記述のうち、正しいものはどれか。

ア　サブディレクトリの下に更にサブディレクトリを置くことができる。
イ　ファイルの下にサブディレクトリを置くことができる。
ウ　ファイルをルートディレクトリの下に置くことはできない。
エ　ルートディレクトリの下に更にルートディレクトリを置くことができる。
オ　ルートディレクトリの下にはサブディレクトリを一つ以上置かなければならない。

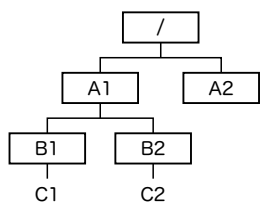
□第１種　平成８年度（1996 年度）春期 午前・問 56

UNIX のファイルシステムの特徴として、当てはまる記述はどれか。

ア　ディレクトリを階層化することができない。
イ　ディレクトリをファイルとして操作することができる。
ウ　一つのファイルに複数の名前を付けることはできない。
エ　ファイル操作をする場合には、レコード長を意識する必要がある。
オ　ファイルの所有者以外によるアクセスを制限することはできない。

■基本　平成 15 年度（2003 年度）秋期 午前・問 35

図の階層型ファイルシステムにおいて、カレントディレクトリが B1 であるとき、ファイル C2 を指す相対パス名はどれか。パス名の表現において、“..”は親ディレクトリを表し、“/”は、パス名の先頭にある場合はルートディレクトリを、中間にある場合はディレクトリ名又はファイル名の区切りを表す。また、図中の□はディレクトリを表すものとする。



ア　../A1/B2/C2　　イ　../B2/C2
ウ　A1/B2/C2　　エ　/A1/B2/C2

■第２種　平成９年度（1997 年度）春期 午前・問 46

15 年秋 35 と同じ。

■第２種　平成 12 年度（2000 年度）春期 午前・問 47

15 年秋 35 と同じ。

■第２種　平成 13 年度（2001 年度）春期 午前・問 35

複数個の同名のディレクトリ A1、A2 が、図の構造で管理されている。各ディレクトリには、ファイル f が存在する。* 印のディレクトリ（カレントディレクトリ）から矢印のディレクトリ配下のファイル f を指定する方法はどれか。

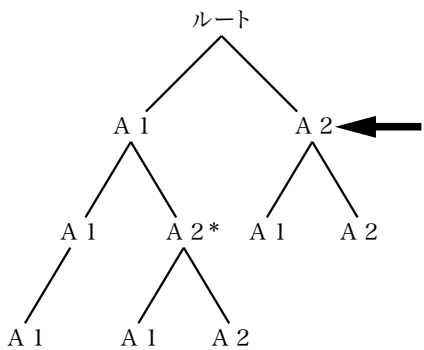
ここで、ファイルの指定方法は、次によるものとする。

ここで、ファイルの指定方法は、次によるものとする。

①ファイルは“ディレクトリ名￥ … ディレクトリ名￥ファイル名”のように、経路上のディレクトリを順に“￥”で区切って指定する。

②カレントディレクトリは“.”で表す。

③１階層上のディレクトリを“..”で表す。



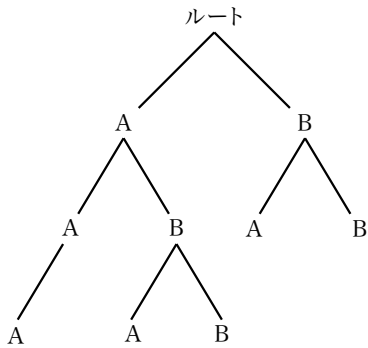
ア　. ￥A2 ￥f　　イ　.. ￥.. ￥A2 ￥f　ウ　.. ￥A1 ￥.. ￥A2 ￥f エ　.. ￥A2 ￥f

■第２種　平成 11 年度（1999 年度）春期 午前・問 50

13 年春 35 と同じ。

■基本　平成 16 年度（2004 年度）春期 午前・問 34

A、B というディレクトリ名をもつ複数個のディレクトリが図の構造で管理されている。



カレントディレクトリを￥A ￥B　→　..　→　.. ￥B　→　. ￥A の順に移動させた場合、最終的なカレントディレクトリはどこか。ここで、ディレクトリの指定は、次の方法によるものとする。

〔ディレクトリ指定方法〕
(1) ディレクトリは“ディレクトリ名 ￥…￥ ディレクトリ名”のように、経路上のディレクトリを“￥”で区切って指定する。
(2)“￥”で始まるときは、左端にルートが指定されているものとする。
(3) カレントディレクトリは“.”で表す。
(4) 1 階層上のディレクトリは“..”で表す。

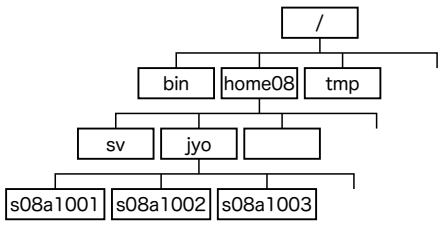
ア　￥A　　イ　￥A ￥A　　ウ　￥A ￥B ￥A　　エ　￥B ￥A

■第２種　平成 12 年度（2000 年度）秋期 午前・問 46

16 年春 34 と同じ

ホームディレクトリ

マルチユーザの利用を想定したUNIXでは、各ユーザごとに専用のディレクトリが用意されているのが、一般的です。そのディレクトリは、ユーザにとっての**ホームディレクトリ**（*home directory*）となります。



2008年入学の情報工学科の学生のホームディレクトリを絶対パスで表すと、次のようになります。

```
/home08/jyo/s08a1001
/home08/jyo/s08a1002
/home08/jyo/s08a1003
    ⋮
```

なお、ログインした直後の**ワーキングディレクトリ（カレントディレクトリ）は、ホームディレクトリとなります。**

▶ Windowsでは、ホームディレクトリは、ドライブHのルートディレクトリである“H:¥”に割り当てられています。

【確認】

ログインしたら、すぐにワーキングディレクトリを表示してみましょう。

```
ipc01%>pwd ↵
/home08/jyo/s08a1001
```

pwdは、ワーキングディレクトリを表示するためのコマンドです。print working directory（直訳すると『ワーキングディレクトリを表示せよ。』）の略です。

ホームディレクトリが、ワーキングディレクトリとなっていることが確認できました。こてで、ディレクトリを表すための記号を復習しておきましょう。

~　ホームディレクトリを表します。

..　親ディレクトリを表します。

.　そのディレクトリ自身を表します。

cdコマンドによって、ワーキングディレクトリを自由に移動することができます。異動先ディレクトリの指定は、絶対パスでも相対パスでも構いません。

なお、パラメータを与えずにcdコマンドを実行すると、ワーキングディレクトリは、ホームディレクトリへと移ります。

▶ Windowsでは、cdでもchdirでも構いません。Windowsでパラメータを与えずにcdまたはchdirコマンドを実行すると、ワーキングディレクトリが表示されます（UNIXのpwdに相当）。

ls は、ファイルおよびディレクトリの一覧を表示するためのコマンドです。list の略です。-l（エル）オプションを付けると、ロング形式（詳細な形式）で表示されます。

▶ Windows で、ファイルとディレクトリの一覧表を表示するのは、DIR コマンドです。詳細な形式で表示したくない場合は /W オプションを与えてワイド形式を指定します。

【演習】

ホームディレクトリのファイル・ディレクトリの一覧を表示します。

```
ipc01%>ls ↵
// 中略（ファイルの一覧が表示されます）
ipc01%>ls -l ↵
// 中略（ファイルの一覧がロング形式で表示されます）
```

リダイレクト（*redirect*：リディレクトとも発音する）を利用すると、コンソール画面に表示される内容を、任意のストリームに出力させることができます。右向き不等号>に続けて、出力ストリームを指定します。

cat コマンドを利用すると、テキストファイルの中身を表示することができます。

▶ Windows では、TYPE コマンドです。

【演習】

ホームディレクトリのファイル・ディレクトリの一覧をロング形式で表示します。いったんファイル list に出力し、それからファイル list の内容を表示します。

```
ipc01%>ls -l > list ↵
ipc01%>cat list ↵
```

mkdir コマンドによってディレクトリを作成し、rmdir コマンドによってディレクトリを削除することができます。

▶ Windows では、MKDIR は MD、RMDIR は RD でも実行できることになっています。

【演習】

ホームディレクトリの直下にディレクトリ TEST を作ってみましょう。

```
ipc01%>cd ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001
ipc01%>mkdir TEST ↵
ipc01%>ls ↵
// 中略（ファイルの一覧が表示されます）
```

作成したディレクトリの詳細を見てみましょう。

```
ipc01%>ls -l ↵

// 中略（ファイルの一覧が表示されます）

drwxr-xr-x  2 s08a1001      512 Jun 20  10:51 TEST/
drwxr-xr-x  2 s08a1001      512 Jun 20  10:52 WWW/
ipc01%>
```

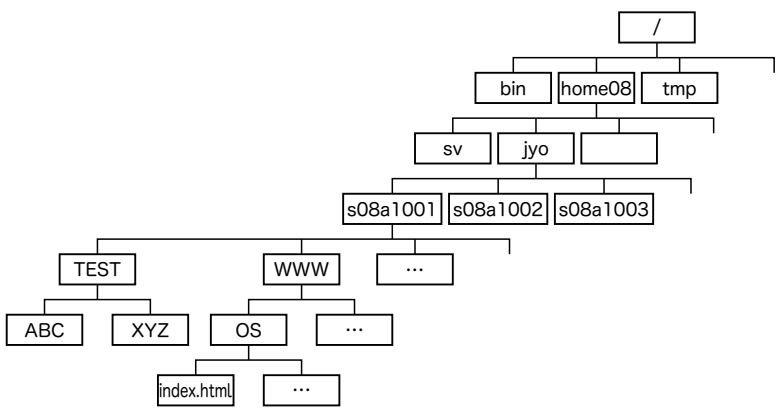
ディレクトリ TEST の下に、ディレクトリ ABC とディレクトリ XYZ を作成しましょう。

```
ipc01%>mkdir TEST/ABC ↵          ※ TEST/ABCは大文字で!!
ipc01%>mkdir TEST/XYZ ↵          ※ TEST/XYZは大文字で!!
```

▶ いったんディレクトリを作成すると、自分で消さない限り消去されませんので、mkdir を２度実行すると、エラーが出ます。

現在、ホームディレクトリの下には、以下のようなサブディレクトリが作られているはずです。

サブディレクトリ WWW
サブディレクトリ WWW/OS
サブディレクトリ TEST
サブディレクトリ TEST/ABC
サブディレクトリ TEST/XYZ



▶ センター側の設定によって、ホームディレクトリ直下の WWW ディレクトリが、以下の URL によって、学内でのみ公開されるようになっています。
http://www.ipc.fit.ac.jp/~s08a****/

【演習】

ホームディレクトリの下の WWW ディレクトリへ移動する方法を確認します。

・絶対パスを利用

```
ipc01%>cd /home08/jyo/s08a1001/WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
```

・ホームディレクトリからの経路を利用

```
ipc01%>cd ~/WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
```

・相対パスを利用（その１）

ワーキングディレクトリがホームディレクトリであるとします。事前に、以下のコマンドを実行します。

```
ipc01%>cd
```

そこから移動します。

```
ipc01%>cd WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
```

・相対パスを利用（その２）

ワーキングディレクトリがTESTであるとします。事前に、以下のコマンドを実行します。

```
ipc01%>cd ~/TEST
```

そこから移動します。

```
ipc01%>cd ../WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
```

・相対パスを利用（その３）

ワーキングディレクトリが TEST/ABC であるとします。事前に、以下のコマンドを実行します。

```
ipc01%>cd ~/TEST/ABC
```

そこから移動します。

```
ipc01%>cd ../../WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
```

【演習】

TEST, TEST/ABC, TEST/XYZ, WWW, WWW/OS を自由に移動してみよう。

【演習】

ホームディレクトリの下のWWW/OSディレクトリへ移動する方法を確認します。

例1

```
ipc01%>cd ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
ipc01%>cd WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
ipc01%>cd OS ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW/OS
```

例2

```
ipc01%>cd ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
ipc01%>cd ~/WWW ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW
ipc01%>cd ./OS ↵
ipc01%>pwd ↵
/home08/jyo/s08a1001/WWW/OS
```

【演習】

WWWディレクトリの下のOSディレクトリにindex.htmlというファイルを作成しているとします（既に作成済みです）。その内容を表示します。

例1　ホームディレクトリからの経路で表す。

```
ipc01%>cat ~/WWW/OS/index.html ↵
```

例2　絶対パスで表す。

```
ipc01%>cat /home08/jyo/s08a1001/WWW/OS/index.html ↵
```

例3　相対パスで表す（ワーキングディレクトリがホームディレクトリであるとする）。

```
ipc01%>cat WWW/OS/index.html ↵
```

※ワーキングディレクトリがホームディレクトリ以外であれば無効。

例4　相対パスで表す（ワーキングディレクトリがWWWディレクトリであるとする）。

```
ipc01%>cat OS/index.html ↵
```

※ワーキングディレクトリが~/WWWディレクトリ以外であれば無効。

例5　相対パスで表す（ワーキングディレクトリがWWW/OSディレクトリであるとする）。

```
ipc01%>cat index.html ↵
```

※ワーキングディレクトリが~/WWW/OSディレクトリ以外であれば無効。

代表的なコマンド

UNIX（Linux）と Windows の代表的なコマンドの概略を示します。

UNIX / Linux のコマンド

clear	画面を消去する。
ls	ファイル／ディレクトリの一覧を表示する。
mkdir	ディレクトリを作成する。
cd	カレントディレクトリを移動する。
pwd	カレントディレクトリを表示する。
rmdir	ディレクトリを削除する。
rm	ディレクトリを削除する。
cat	ファイルの内容を表示する。
cp	ファイルをコピーする。
mv	ファイルを移動する／ファイル名を変更する。
man	コマンドの説明を表示する。

Windows のコマンド

cls	画面を消去する。
chdir	カレントディレクトリを移動する。
cd	カレントディレクトリを移動する。
mkdir	ディレクトリを作成する。
md	ディレクトリを作成する。
rmdir	ディレクトリを削除する。
rd	ディレクトリを削除する
type	ファイルの内容を表示する。
copy	ファイルをコピーする。
move	ファイルを移動する／ファイル名を変更する。
rename	ファイル名を変更する。
ren	ファイル名を変更する。
help	コマンドの説明を表示する。

【演習】

Windows Vista の ROBOCOPY コマンドについて調べてみよう。

主記憶管理

主記憶（*main memory*）の管理は、OSの重要な役割の一つです。ここでは、主記憶管理について学習します。

> **JIS X0011 01.16【主記憶（装置）】の定義**
> **命令及びその他のデータを、実行又は処理に先立ってロードしなければならない内部記憶装置。＜備考＞主記憶装置という用語は、大型の計算システムの場合によく使われる。**

プログラムのロード

プログラムは、実行されるときに、それを格納した磁気ディスクなどの補助記憶装置から主記憶装置へと読み込まれます。読み込みを行うことを、**ロード**（*load*）するといい、ロードを行うプログラムを**ローダ**（*loader*）と呼びます。

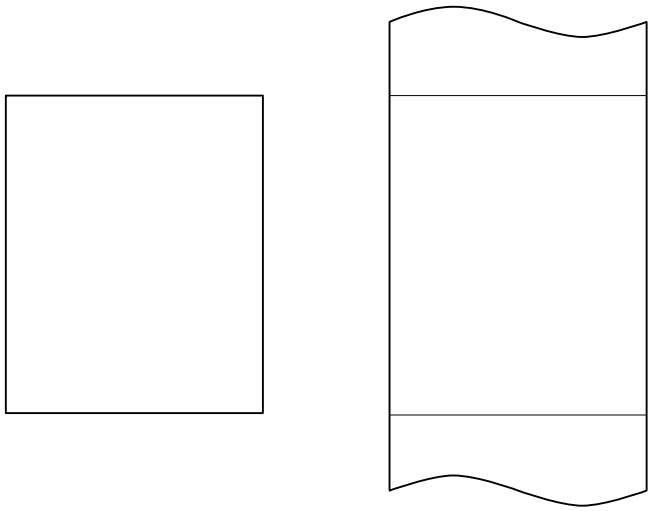
> **JIS X0007 05.04【ローダ】の定義**
> **外部記憶装置から内部記憶装置に他のプログラムをコピーしたり、外部記憶装置から内部記憶装置、又は内部記憶装置からレジスタにデータをコピーしたりするプログラム。**

主記憶の空いているところにプログラムをロードできると都合が良いので、通常のプログラムは、任意のアドレスに配置できるようになっています。そのように実現されているプログラムは、**再配置可能プログラム**（*relocatable program*）と呼ばれます。再配置可能プログラムを主記憶に読み込んだ際に、ロード位置に対応してプログラム内のアドレス情報を補正することを、**再配置**（*relocation*）といいます。

プログラムの実行直前に配置して実行中は位置を変更しない方式を**静的再配置**と呼び、プログラムの実行中にダイナミックに配置されるものを**動的再配置**と呼びます。

> **JIS X0007 05.14【再配置】の定義**
> **主記憶装置のどの部分にでもロードできる目的プログラムの全体又は一部分に関する用語。＜備考＞先頭アドレスは、ローダが確立する。次に、ローダは、プログラムの各部分をロードした記憶場所に合わせて、アドレスを補正する。**

▶ ベースレジスタ方式による再配置では、再配置可能なロードモジュールを主記憶装置にロードして、その先頭番地をベースレジスタに格納します。また、命令のアドレス部には、その先頭アドレスからの相対アドレスを格納します。ベースレジスタの値と、相対アドレスを加算して、有効アドレスを決定します。



■基本　平成 19 年度（2007 年度）秋期 午前・問 29

プログラムを実行するために主記憶に読み込んだとき、ロード位置に対応してプログラム内のアドレス情報を補正することを示す用語はどれか。

ア　再コンパイル　　イ　最適化　　ウ　再配置　　エ　リロード

■第２種　平成 12 年度（2000 年度）秋期 午前・問 35

プログラムをロードして実行するとき、プログラムのロード位置に対応してプログラム内のアドレスを補正することを示す用語はどれか。

ア　仮想記憶　　イ　最適化　　ウ　再配置　　エ　リンキング

■第２種　平成８年度（1996 年度）秋期 午前・問 34

プログラムをロードして実行するとき、プログラムのロード位置に対応してプログラム内のアドレスを補正することを示す用語はどれか。

ア　仮想記憶　　イ　最適化　　ウ　再配置　　エ　翻訳　　オ　連係編集

□第１種　平成８年度（1996 年度）午前・問 27

再配置可能なロードモジュールのロードと実行に、最も関係の深いレジスタはどれか。

ア　インデックスレジスタ　イ　シフトレジスタ　ウ　汎用レジスタ
エ　ベースレジスタ　オ　命令レジスタ

動的リンキングと動的リンクライブラリ（DLL）

個別に作成したプログラム・モジュールを結合して、一つプログラムにすることを**連係編集＝リンキング**（*linking*）といいます。プログラムの実行前に、**連係編集プログラム＝リンカ**（*linker*）**＝リンケージ・エディタ**（*linkage editor*）を用いてリンクするのが**静的（スタティック）リンキング**で、プログラムの実行中にリンクするのが**動的（ダイナミック）リンキング**です。

オペレーティングシステムの本体は巨大であり、すべてを主記憶にロードすることはできません。したがって、一部をライブラリの形で用意しておき、必要なときにのみロードできる動的リンキングの機能を利用すると、記憶域を大幅に節約できます。

また、オペレーティングシステム上で動作するプログラムから共通して利用するようなライブラリも、それぞれのプログラムで同じものを個別に持つのではなく、必要に応じて主記憶にロードするようにしておくと、各プログラムでライブラリをもつ必要がなくなりますから、実行プログラムの大きさを抑えることもできます。

以上のような目的を実現するのために、動的リンキングを前提としたソフトウエア・モジュール群をまとめたライブラリ・ファイルが、**DLL**（*dynamic linking library*）**＝動的リンク・ライブラリ**です。

モジュールが呼び出された時点で、OSがハードディスクなどの補助記憶域装置から主記憶へと読み出してリンクします。

MS-Windowsであれば、SystemやSystem32フォルダの中に.DLLという拡張子のついたファイルがたくさんあります。これらが動的リンクライブラリです。

■基本　平成 16 年度（2004 年度）春期 午前・問 45

プログラムを構成するモジュールの結合を、プログラムの実行時に行う方式はどれか。

ア　インタプリタ　　イ　オーバレイ
ウ　静的リンキング　　エ　動的リンキング

■基本　平成 15 年度（2003 年度）春期 午前・問 43

動的リンキングの機能はどれか。

ア　プログラム実行時に、共用ライブラリやシステムライブラリのモジュールをロードする。
イ　プログラム実行時に、適切なアドレスに目的プログラムをロードする。
ウ　プログラム実行時に、読み込まれたページの論理アドレスを物理アドレスに変換する。
エ　プログラムの実行に先立って、複数の目的プログラムを連係編集（リンケージエディット）する。

■基本　平成 20 年度（2008 年度）秋期 午前・問 38

動的リンクライブラリ（DLL）の特徴として、適切なものはどれか。

ア　アプリケーションがメモリにロードされるときに、同時にリンカによって組み込まれる。
イ　アプリケーションの実行中、必要になったときに OS によって連係される。
ウ　コンパイル時に、コンパイラによってアプリケーションに組み込まれる。
エ　コンパイルの前に、プリコンパイラによってアプリケーションに組み込まれる。

■第２種　平成９年度（1997 年度）春期 午前・問 38

動的リンクライブラリ（DLL）に関する正しい記述はどれか。

ア　コンパイル時に、コンパイラによって組み込まれる。
イ　コンパイル前に、プリコンパイラによって組み込まれる。
ウ　実行時に、オペレーティングシステムによって連係される。
エ　ロードモジュール作成時に、連係編集プログラムによって連係され組み込まれる。

■第２種　平成 11 年度（1999 年度）秋期 午前・問 36

動的リンクライブラリ（DLL）に関する記述として、適切なものはどれか。

ア　コンパイル時に、コンパイラによって組み込まれる。
イ　コンパイルの前に、プリコンパイラによって生成される。
ウ　実行時に、OS によって連係される。
エ　ロードモジュール作成時に、連係編集プログラムによって連係され組み込まれる。

□第１種　平成 12 年度（2000 年度）午前・問 31

動的リンキングに関する記述として、適切なものはどれか。

ア　仮想記憶方式のコンピュータにおいて、読み込まれたページの論理アドレスを物理アドレスに対応させる。
イ　プログラム実行時に、共用ライブラリやシステムコールライブラリのモジュールをロードする。
ウ　プログラム実行時に、適切なアドレスにロードする。
エ　プログラム実行時に、ヒープ上に可変長データのための領域を獲得する。

□第１種　平成８年度（1996 年度）午前・問 22

オペレーティングシステムに関する記述のうち、正しいものはどれか。

ア　実行中のプログラムを主記憶装置の異なった領域で実行するために、実行中に新たな絶対アドレスを割り当てる処理を、動的再配置という。
イ　主記憶の利用効率を向上させる一つの方法して、スワッピングがあり、主記憶から外部記憶へプログラムを移すことをスワップイン、その逆をスワップアウトという。
ウ　一つのプログラムのサブモジュールを複数のプロセッサ上で並列に実行することを、マルチプログラミングという。
エ　プログラムの一部をユーザプログラムによって主記憶装置の中の同じ区域にロードし、実行することを、ダイナミックリンケージという。
オ　リンカ（連係編集プログラム）が出力し、かつオペレーティングシステムのもとで実行可能なプログラムを、オブジェクトモジュールという。

再入可能（リエントラント）

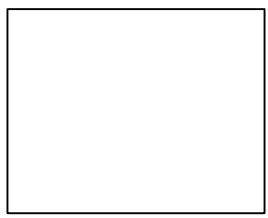
再入可能（*reentrant*）とは、マルチタスク環境で、複数のタスクから同時に同一コードを実行できるようにされたプログラムの性質のことです。

> **JIS X0007 03.20【再入可能】の定義**
> **プログラムの実行可能な版の全体又は一部分に関する用語であって、繰り返して入ることも、直前の実行が完了しないうちに入ることもでき、しかもそのようなプログラムの毎回の実行が他の実行とは無関係である場合に使用する。**

▶　エントランスは≪入り口≫という意味ですね。英語の re は、再びという意味です（リプレイ＝再生／リコンストラクション＝再構築）。

あるタスクから呼び出されたリエントラントなプログラムが実行中に他のタスクから呼び出された場合、それぞれのタスクに対して正しい結果を返すことになります。

再入可能プログラムを実現するためには、プログラムを手続き部分とデータ部分に分割して、プロセスごとにデータ部分をスタックとしてもつ必要があります。タスクごとに固有のリソースだけを変更します。共有のアドレス領域の変更などは許されません。



■基本　平成 17 年度（2005 年度）春期 午前・問 36

処理が終了していないプログラムが、別のプログラムから再度呼び出されることがある。このプログラムが正しく実行されるために備えるべき性質はどれか。

ア　再帰的（リカーシブ）　　イ　再使用可能（リユーザブル）
ウ　再入可能（リエントラント）　　エ　再配置可能（リロケータブル）

■基本　平成 16 年度（2004 年度）春期 午前・問 42

複数のプロセスから同時に呼び出されたときに、互いに干渉することなく並行して処理することができるプログラムの性質を表すものはどれか。

ア　リエントラント　　イ　リカーシブ
ウ　リユーザブル　　エ　リロケータブル

■基本　平成 15 年度（2003 年度）春期 午前・問 41

あるプログラムモジュールが、複数のタスクの要求に応じて同時・並行的に実行可能である場合、この性質を何というか。

ア　再使用可能　　イ　再入可能
ウ　静的再配置可能　　エ　動的再配置可能

■基本　平成 14 年度（2002 年度）春期 午前・問 41

リアルタイムシステムにおいて、複数のタスクから並行して呼び出される共用ライブラリのプログラムに要求される性質として、適切なものはどれか。

ア　リエントラント　　イ　リカーシブ
ウ　リユーザブル　　エ　リロケータブル

■第２種　平成 12 年度（1998 年度）秋期 午前・問 36

処理が終了していないプログラムが、別のプログラムから再度呼び出されることがある。このプログラムが正しく実行されるために備えるべき性質はどれか。

ア　再帰的（リカーシブ）　　イ　再使用可能（リユーザブル）
ウ　再入可能（リエントラント）　　エ　再配置可能（リロケータブル）

■第２種　平成 10 年度（1998 年度）春期 午前・問 40

処理が終了していないプログラムが、別のプログラムから再度呼び出されることがある。このプログラムが正しく実行されるために備えるべき性質はどれか。

ア　再帰的（リカーシブ）　　イ　再入可能（リエントラント）
ウ　再配置可能（リロケータブル）　　エ　再利用可能（リユーザブル）

■第２種　平成８年度（1996 年度）春期 午前・問 34

次の記述中の［　　］に入れるべき適切な字句の組合せはどれか。

主記憶にロードされた一つの［ a ］を複数の［ b ］で共用して並行に実行するためには、その［ a ］が［ c ］である必要がある。

	a	b	c
ア	サブルーチン	タスク	再帰的
イ	サブルーチン	プログラム	再帰的
ウ	タスク	サブルーチン	再帰的
エ	タスク	プログラム	再入可能
オ	プログラム	タスク	再入可能

■基本　平成 13 年度（2001 年度）秋期 午前・問 41

プログラムの構造に関する記述のうち、適切なものはどれか。

ア　再帰的処理のためには、実行途中の状態を FIFO 方式で記録し、制御する必要がある。
イ　再入可能プログラムを実現するためには、プログラムを手続き部分とデータ部分に分割して、データ部分をプロセスごとにもつ必要がある。
ウ　逐次再使用可能なプログラムは、再入可能でもある。
エ　複数のプロセスで同時に実行できるようにしたプログラムは、再帰的である。

■第２種　平成 12 年度（2000 年度）春期 午前・問 38

再入可能プログラムの特徴として、適切なものはどれか。

ア　実行時に必要な手続きを補助記憶装置から呼び出しながら動作する。実行時の主記憶領域の大きさに制限があるときに、有効な手法である。
イ　手続きの内部から自分自身を呼び出すことができる。
ウ　複数のタスクで利用するとき、一時に一つのタスクでの使用が可能である。
エ　複数のタスクによって並行して実行されても、それぞれのタスクに正しい結果を返す。

■第２種　平成 11 年度（1999 年度）春期 午前・問 35

再入可能（リエントラント）プログラムの説明として、最も適切なものはどれか。

ア　一度実行した後、ロードし直さずに再び実行を繰り返しても、正しい結果が得られる。
イ　実記憶上のどこのアドレスに配置しても実行することが可能である。
ウ　複数のセグメントに分割されており、セグメント単位にロードして実行することが可能である。
エ　複数のタスクで並行して実行しても、正しい結果が得られる。

■第２種　平成９年度（1997 年度）春期 午前・問 35

再入可能（リエントラント）プログラムの説明として、最も適切なものはどれか。

ア　一度実行した後、ロードし直さずに再び実行を繰り返しても、正しい結果が得られる。
イ　主記憶上のどこのアドレスに配置しても実行することが可能である。
ウ　同時に複数のタスクが共有して実行しても、正しい結果が得られる。
エ　複数のセグメントに分割されており、セグメント単位にローディングして実行することが可能である。

■第２種　平成７年度（1995 年度）秋期 午前・問 37

再入可能（リエントラント）プログラムの説明として、最も適切なものはどれか。

ア　一度実行した後、再ロードすることなく再び実行を繰り返しても、正しく実行が可能

である。
イ　実行中であっても、重ねて同一プログラムの実行が可能である。
ウ　主記憶上のどこのアドレスに配置しても実行が可能である。
エ　障害によって実行が中断した場合、チェックポイントにおいて記録したデータを使用して実行の再開が可能である。
オ　複数のセグメントに分割し、セグメント単位にローディングして実行が可能である。

□第１種　平成９年度（1997 年度）午前・問 25

ある手続きが複数のプロセスによって並行して実行されるとき、それぞれのプロセスに正しい結果を返すことができる場合、この手続きの性質を何というか。

ア　再帰的　　イ　再入可能　　ウ　再配置可能　　エ　逐次再使用可能

□第１種　平成８年度（1996 年度）午前・問 30

再入可能（リエントラント）プログラムに関する記述のうち、正しいものはどれか。

ア　再入可能プログラムでは、変数をタスク単位に格納しなければならない。
イ　再入可能プログラムは、Ｃ言語では作成できない。
ウ　再入可能プログラムは、逐次使用可能プログラムから呼び出すことはできない。
エ　実行途中で待ち状態が発生するプログラムは、再入可能ではない。
オ　逐次再使用可能なプログラムは、再入可能プログラムとして使用できる。

□第１種　平成 10 年度（1998 年度）午前・問 28

再入可能（リエントラント）プログラムに関する記述のうち、正しいものはどれか。

ア　再入可能プログラムは、逐次再使用可能プログラムから呼び出すことはできない。
イ　再入可能プログラムは、局所変数をタスク単位に格納しなければならない。
ウ　実行途中で待ち状態が発生するプログラムは、再入可能ではない。
エ　逐次再使用可能なプログラムは、再入可能プログラムとして使用できる。

□第１種　平成 11 年度（1999 年度）午前・問 28

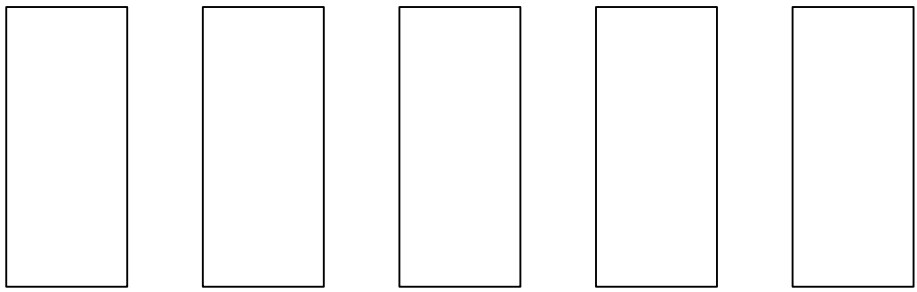
リアルタイムシステムにおいて、複数のタスクから並行して呼び出される共用ライブラリのプログラムに要求される性質として、適切なものはどれか。

ア　リエントラント　イ　リカーシブ　　ウ　リユーザブル　　エ　リロケータブル

フラグメンテーション

記憶領域の割当てと解放を繰り返すことによって、こま切れの未使用領域が多数できてしまうことを**フラグメンテーション**（*fragmentation*）といいます。フラグメンテーションの発生によって、合計としては十分な空きメモリ領域があるのに、必要とするメモリ領域を獲得できなくなることがあります。

フラグメンテーションによる空き領域を詰めることを、**コンパクション**（*compaction*）と呼びます。



■基本　平成 13 年度（2001 年度）秋期 午前・問 33

フラグメンテーションに関する記述のうち、適切なものはどれか。

ア　可変長ブロックのメモリプール管理方式では、いろいろな大きさのメモリ領域の獲得や返却を行ってもフラグメンテーションは発生しない。
イ　固定長ブロックのメモリプール管理方式では、可変長ブロックのメモリプール管理方式よりもメモリ領域の獲得と返却を速く行えるが、フラグメンテーションが発生しやすい。
ウ　フラグメンテーションの発生によって、合計としては十分な空きメモリ領域があるのに、必要とするメモリ領域を獲得できなくなることがある。
エ　メモリ領域の獲得と返却の頻度が高いシステムでは、メモリ領域返却のたびにガーベジコレクションを行う必要がある。

■第２種　平成９年度（1997 年度）春期 午前・問 34

オペレーティングシステムが、記憶領域の割当てと解放を繰り返すことによって、こま切れの未使用領域が多数できてしまう場合がある。この現象を何というか。

ア　コンパクション　　イ　スワッピング
ウ　フラグメンテーション　　エ　ページング

■第２種　平成 12 年度（2000 年度）春期 午前・問 36

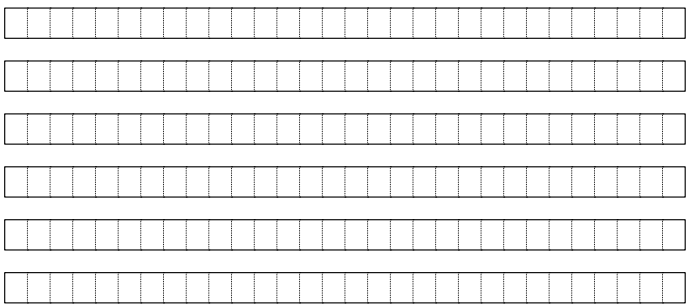
OS が記憶領域の割当てと解放を繰り返すことによって、こま切れの未使用領域が多数できてしまう場合がある。この現象を何というか。

ア　コンパクション　　イ　スワッピング
ウ　フラグメンテーション　　エ　ページング

ガーベジコレクション

プログラムが使用しなくなったヒープ領域を回収して再度使用可能にする処理が、**ガーベジコレクション**（*garbage collection*）です。

```
A$ = "ABC"
B$ = "123"
A$ = "ABCDEF"
B$ = "1234567"
A$ = "ABCDEFGHIJK"
```



■基本　平成 16 年度（2004 年度）秋期 午前・問 32

記憶領域の動的な割当て及び解放を繰り返すことによって、どこからも利用されない記憶領域が発生することがある。このような記憶領域を再び利用可能にする処理はどれか。

ア　ガーベジコレクション　　イ　スタック
ウ　ヒープ　　エ　フラグメンテーション

□第 1 種　平成 10 年度（1998 年度）午前・問 27

プログラム実行時のメモリ管理に関する記述として、正しいものはどれか。

ア　主記憶域の容量をこえるプログラムを実行できるようにすることを、メモリコンパクションという。
イ　プログラムが使用しなくなったヒープ領域を回収して再度使用可能にする処理を、ガーベジコレクションという。
ウ　プログラム実行中必要になった時点で、ライブラリに用意されているプログラムモジュールをロードする方法を、動的再配置という。
エ　メモリを有効に利用するために、オブジェクトモジュールを主記憶上で移動させ、プログラムを実行することを、動的リンキングという。

オーバレイ

ロードモジュールを、同時に実行されることのない複数個の部分（セグメント）に分割し、実行時にセグメントを交互にローディングする方式を、**オーバレイ**（*overlay*）と呼びます。

```
abcd() { /*・・・*/ }
xy()  { /*・・・*/ }
z()   { /*・・・*/ }

xyz()
{
    xy();
    z();
}

int main(void)
{
    abcd();
    xyz();
}
```

■第２種　平成７年度（1995 年度）春期 午前・問 31

ロードモジュールを、同時に実行されることのない複数個の部分（セグメント）に分割し、実行時にセグメントを交互にローディングする方式はどれか。

ア　オーバレイ　　イ　スワッピング　　ウ　ダイナミックリンク
エ　動的再配置　　オ　ロールインロールアウト

■第２種　平成 12 年度（2000 年度）春期 午前・問 37

ロードモジュールを、排他的に実行される複数個の部分（セグメント）に分割し、実行時に必要なセグメントをローディングする方式はどれか。

ア　オーバレイ　　イ　スワッピング
ウ　動的再配置　　エ　動的リンキング

スワッピング

プログラムの優先度に応じて、コンピュータ上の主記憶装置の内容と補助記憶装置の内容を入れ替えて処理を行うのが**スワッピング**（*swapping*）です。

> **JIS X0010 05.09【スワッピングの定義】**
> **主記憶装置の領域の内容と補助記憶装置の領域の内容とを交換する処理。**

たとえば、新しいプログラムを実行する場合、主記憶装置上に必要な領域が存在しなければ、優先順位の低いプログラムやデータを主記憶装置から補助記憶装置上のスワップエリアに退避させて必要な領域を確保して、その確保した領域にプログラムをロードして実行させる、といったことを行います。

主記憶装置の内容を退避させるのが**スワップアウト**で、逆に補助記憶装置から主記憶装置に移すのが**スワップイン**です。

■第２種　平成 10 年度（1998 年度）秋期 午前・問 36

あるプログラムの実行中に、それより優先度の高いプログラムを実行するために、もとのプログラムを補助記憶装置に移して優先度の高いプログラムをロードする処理はどれか。

ア　オーバレイ　　イ　スワッピング　　ウ　ページング　　エ　リロケーション

仮想記憶

一般に、主記憶装置の容量は、ユーザが実行する全てのプログラムを一度に格納するには十分ではありません。補助記憶装置などを用いて、実際の主記憶装置の容量以上のアドレス空間を仮想的に作り出す**仮想記憶**（*virtual storage*）を採用します。仮想記憶を利用すると、主記憶装置の容量を意識しなくて済むことになります。

> **JIS X0010 05.11 の定義【仮想記憶の定義】**
> **計算機システムの利用者にとってアドレス可能な主記憶装置とみなしうる記憶空間であって、仮想アドレスが実アドレスへ写像されるもの。〈注〉仮想記憶の大きさは、計算機システムのアドレス指定方式及び使用可能な補助記憶装置の量によって制限され、主記憶場所の実際の量によっては制限されない。**

最初に、仮想記憶の考え方を、たとえ話で理解しましょう。

まず、主記憶装置を『テーブル』と考えることにします。そのテーブルに 10 枚のカードしか置けないとします。ここで、テーブルの横に、100 枚のカードを置ける『補助テーブル』を用意します。（マジシャンのように一般人に気付かれないように巧みに）テーブルと補助テーブル上のトランプを入れかえれば、あたかも『テーブル』上で 10 枚ではなく 100 枚のカードをテーブル上で扱えるようになります。

▶ テーブルを使うユーザに気付かれないように、マジシャンである OS がカードの入れ替え作業を行います。カードのことをページと呼びます。

『テーブル』と『補助テーブル』間で素早くデータをやりとりしなければなりません。そのため補助テーブルである補助記憶装置は、高速かつ大容量でなければなりませんから、通常は**ハードディスク**が使われます。

■第２種　平成 10 年度（1998 年度）秋期 午前・22

仮想記憶方式を用いているコンピュータシステムにおいて、実記憶に格納しきれないプログラムやデータを格納するための装置として、適切なものはどれか。

ア　CD–ROM　　イ　VRAM
ウ　キャッシュメモリ　　エ　ハードディスク

■第２種　平成 8 年度（1996 年度）秋期 午前・問 23

仮想記憶方式を用いているシステムにおいて、実記憶に格納しきれないプログラムやデータを格納するための装置として最も適切なものはどれか。

ア CD–ROM　　イ VRAM　　ウ キャッシュメモリ
エ ハードディスク　　オ フロッピーディスク

仮想記憶の基本的な考え方は、以下のようになっています。

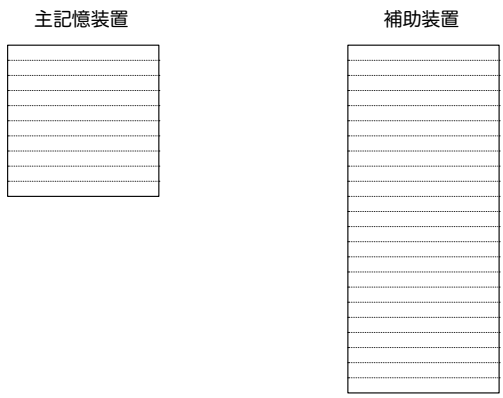
・主記憶装置を一定の大きさの区画に分割します。この区画をページ枠といいます。
・プログラムはいったん、補助記憶装置の外部ページ記憶域と呼ばれる領域に記憶されます。
・外部ページ記憶城は，ページ枠と同じ大きさの**スロット**（*slot*）と呼ばれる区画に分割されます。すなわち、外部ページ記憶域に記憶されたプログラムは自動的にスロットの大きさに分割されることになります。
・ページ枠やスロットに記憶されているプログラムを**ページ**（*page*）といいます。一つのページは一般に２キロバイト（あるいは４キロバイトや８キロバイトなど）の大きさです。当然、ページ枠もページと同じ大きさです。
・主記憶装置と補助記憶装置の外部ページ記憶城を**論理アドレス空間**（*logical address space*）といいます・
・外部ページ記憶城に記憶されているプログラムのうち，実行に必要なスロットのページを主記憶装置の空いているページ枠に移動して実行します。

このように，仮想記憶方式ではページ単位で外部ページ記憶域に記憶されているプログラムを主記憶装置のページ枠に移して実行を繰り返します。その際、主記憶装置にプログラムを移すことをロードといいます。

ページング

仮想記憶をページという単位で格納し、実記憶をページ枠で管理する方式がページングです。**実記憶領域の利用効率が高く、領域管理も容易です**。

補助記憶装置の外部ページ記憶城にあるスロットを主記憶装置のページ枠に移すことを**ページイン**（*page-in*）、逆に実行の終わったページをスロットに移すことを**ページアウト**（*page-out*）といいます。



ページサイズを小さくすると、無駄な領域が少なくなって資源の利用効率は向上しますが、アドレス変換の頻度は増大します。一方、ページサイズを大きくすると、無駄な領域が多くなってスワッピングの頻度が増大します。

■第２種　平成 11 年度（1999 年度）春期 午前・問 19

記憶空間を一定の大きさに区切って管理し、仮想記憶を実現する方式はどれか。

ア　スラッシング　　イ　スワッピング　　ウ　ブロッキング　　エ　ページング

■第２種　平成８年度（1996 年度）春期 午前・問 33

記憶空間を一定の大きさに区切って管理し、仮想記憶を実現する方式を何というか。

ア　スラッシング　　イ　スワッピング　　ウ　ブロッキング
エ　ページング　　オ　ローディング

■基本　平成 19 年度（2007 年度）春期 午前・問 28

仮想記憶方式の一つに、仮想アドレス空間を固定長の領域に分割して管理するものがある。この固定長の領域を示す用語はどれか。

ア　セクタ　　イ　セグメント　　ウ　フレーム　　エ　ページ

■基本　平成 16 年度（2004 年度）春期 午前・問 30

仮想記憶方式の一つに、仮想アドレス空間を固定長の領域に分割して管理するものがある。この固定長の領域を示す用語はどれか。

ア　セクタ　　イ　セグメント　　ウ　フレーム　　エ　ページ

■第２種　平成 11 年度（1999 年度）秋期 午前・問 31

仮想記憶方式の一つに、仮想アドレス空間を固定長の領域に分割して管理するものがある。この固定長の領域を示す用語はどれか。

ア　セクタ　　イ　セグメント　　ウ　フレーム　　エ　ページ

■第２種　平成９年度（1997 年度）秋期 午前・問 33

仮想記憶方式の一つに、仮想アドレス空間を固定長の領域に分割して管理するものがある。この固定長の領域を示す用語はどれか。

ア　セクタ　　イ　フレーム　　ウ　ページ　　エ　モジュール

多重プログラミング方式では、ページの入替えが頻繁に起こる場合があります。これを**スラッシング**（*thrashing*）といいます。

一つのプロセス（タスク）を実行するために必要となるメモリの総量のことを**ワーキングセット**（*working set*）と呼びます。ワーキングセットが大きい場合、主記憶装置と補助記憶装置との間で頻繁にスラッシングが発生することになります。プロセスごとのワーキングセットをできるだけ小さくしておくことが望ましいと考えられます。

スラッシング発生時は、プログラムの実行よりもページングに多くの時間が必要となります。

■基本　平成 15 年度（2003 年度 ）春期 午前・問 30

ページング方式の仮想記憶システムにおいて、スラッシングが発生しているときの状況はどれか。

	アプリケーションの CPU 使用率	主記憶と補助記憶の間のページ転送量
ア	高い	多い
イ	高い	少ない
ウ	低い	多い
エ	低い	少ない

■第２種　平成 10 年度（1998 年度）春期 午前・問 35

ページング方式を用いて仮想記憶を実現しているシステムにおいて、スラッシングが発生しているときの状況はどれか。

ア　CPU の利用効率は高く、主記憶と補助記憶との間のページ転送量は多い。
イ　CPU の利用効率は高く、主記憶と補助記憶との間のページ転送量は少ない。
ウ　CPU の利用効率は低く、主記憶と補助記憶との間のページ転送量は多い。
エ　CPU の利用効率は低く、主記憶と補助記憶との間のページ転送量は少ない。

□第１種　平成 8 年度（1996 年度）午前・問 24

ページング方式を用いた仮想記憶システムの主記憶管理方式に関係の深い用語を二つ選べ。

ア　オーバレイ
イ　スラッシング
ウ　メモリコンパクション
エ　メモリフラグメンテーション
オ　ワーキングセット

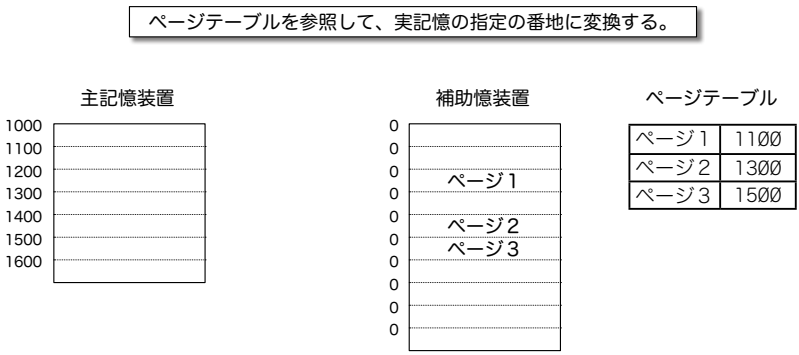
アドレス変換

ページングで問題になるのは、主記憶装置のページインの番地がわからないことです。仮想記憶方式では、空いたページ枠が発生すると、次に実行するページをページインするだけなので、ページ枠の番地に合わせて命令の番地を変換する必要があります。これが**アドレス変換**（*address translation*）です。

外部ページ記憶領域に記憶されているプログラムの、各命令につけられている番地を**静的アドレス**といい、アドレス変換をして主記憶装置のページ枠に記憶されたときの番地を**動的アドレス**といいます。

アドレス変換の代表的な方法が、**DAT＝動的アドレス変換機構**（*dynamic address translation*）という、ハードウェアを用いて行う方法です。

DATは、ページインした命令を実行する時点でアドレス変換を行います。外部ページ記憶域の番地はページ単位に0番地から始まっているのに対し、ページインするページの番地はページテーブルを参照して、動的アドレスに変換されます。



■第２種　平成 10 年度（1998 年度）秋期 午前・問 39

動的アドレス変換の説明として、最も適切なものはどれか。

ア　仮想記憶システムにおいて、仮想アドレスから実アドレスへの変換を行うこと。
イ　実行中のプログラムを移動して新しい場所で実行できるように、プログラムの基底アドレスを変更すること。
ウ　主記憶に対する読み書きを、キャッシュメモリで代行すること。
エ　プログラムの実行途中にモジュールを追加するため、モジュール間のアドレス参照を解決すること。

セグメンテーションページング

論理的につながりのあるページの集まりをセグメントといいます。**セグメンテーションページング**（*segnmentation paging*）は，ページインとページアウトをセグメント単位に行います。通常のページング方式に比べて，ページング回数が少なくなる傾向にあります。

ページリプレースメント

実記憶上にないページをアクセスした場合の処理は、次のようになります。

ページフォールト ⇨ 置換え対象ページの決定 ⇨ ページアウト ⇨ ページイン

すなわち、以下のようになります。

①ページフォールト：必要なページが主記憶に存在しないことに気付きます。
②置換え対象ページの決定：主記憶上の不要なページを選びます。
③ページアウト：②で決定したページを補助記憶に退避させます。
④ページイン：必要なページを補助記憶から主記憶に読み込みます。

プロセスの多重度を上げると、実メモリの要求が増えるため、ページフォールトの発生が増加します。

■基本　平成 20 年度（2008 年度）秋期 午前・問 27

ページング方式の仮想記憶において、主記憶に存在しないページをアクセスした場合の処理や状態の順番として、適切なものはどれか。ここで、主記憶には現在、空きのページ枠はないものとする。

ア　置換え対象ページの決定→ページイン→ページフォールト→ページアウト
イ　置換え対象ページの決定→ページフォールト→ページアウト→ページイン
ウ　ページフォールト→置換え対象ページの決定→ページアウト→ページイン
エ　ページフォールト→置換え対象ページの決定→ページイン→ページアウト

■第２種　平成 12 年度（2000 年度）春期 午前・問 32

ページング方式の仮想記憶において、実記憶上にないページをアクセスした場合の処理と状態の順番として、適切なものはどれか。ここで、実記憶には現在、空きページはないものとする。

ア　置換え対象ページの決定 → ページアウト → ページフォールト → ページイン
イ　置換え対象ページの決定 → ページイン → ページフォールト → ページアウト
ウ　ページフォールト → 置換え対象ページの決定 → ページアウト → ページイン
エ　ページフォールト → 置換え対象ページの決定 → ページイン → ページアウト

ページアウトする置き換え対象ページを決定する際は、**『その時点以降の最も遠い将来まで参照されないのがどのページであるのか』**を推測することになります。

基本的な考え方として、以下の二つがあります。

・FIFO（First–In First–Out）

主記憶域に最初にページインしたページを最初にページアウトします。すなわち、**もっとも長いあいだ主記憶域に存在しているページをページアウトする**方式です。

・LRU（Least Recently Used）

最後に使われてから最も時間が経過しているページを最初にページアウトします。すなわち、**最も長い間参照されていないページをページアウトする**方式です。

■基本　平成 13 年度（2001 年度）秋期 午前・問 30

仮想記憶システムで使用されるページ置換えアルゴリズムには、FIFO 方式や LRU 方式などがある。これらのページ置換えアルゴリズムの基本的な考え方として、適切なものはどれか。

ア　その時点以降に参照される頻度が最も高いページがどれかを推測する。
イ　その時点以降に参照される頻度が最も低いページがどれかを推測する。
ウ　その時点以降の最も近い将来に参照されるページがどれかを推測する。
エ　その時点以降の最も遠い将来まで参照されないページがどれかを推測する。

■基本　平成 17 年度（2005 年度）秋期 午前・問 27

仮想記憶方式でページフォールトが発生したとき、主記憶に最も古くから存在するページを追い出すアルゴリズムはどれか。

ア　FIFO（First–in First–out）　　イ　LFU（Least Frequently Used）
ウ　LIFO（Last–in First–out）　　エ　LRU（Least Recently Used）

■基本　平成 18 年度（2006 年度）春期 午前・問 27

仮想記憶におけるページ置換えアルゴリズムとしてFIFO方式を採用する。主記憶のページ枠が３で、プログラムが参照するページ番号の順序が、４→３→２→１→３→５→２のとき、ページインは何回行われるか。ここで、初期状態では、主記憶には何も読み込まれていないものとする。

ア　２　　イ　３　　ウ　５　　エ　６

□第１種　平成 10 年度（1998 年度）午前・問 26

仮想記憶管理のページ置換えアルゴリズムとして FIFO を用いる。実記憶のページ枠が 3 ページ分で、参照する仮想ページの番号の順が、4, 3 ,2 ,1, 4, 3, 5, 4, 3, 2, 1, 5 のとき、ページインは何回行われるか。ここで、初期状態では、実記憶には何も読み込まれていないものとする。

ア　7　　イ　8　　ウ　9　　エ　10

■基本　平成 16 年度（2004 年度）秋期 午前・問 30

仮想記憶管理のページ入替え方式のうち、最後に使われてからの経過時間が最も長いページを入れ替えるものはどれか。

ア　FIFO　　イ　LFU　　ウ　LIFO　　エ　LRU

■第２種　平成９年度（1997 年度）春期 午前・問 19

仮想記憶のページ入替方式のうち、使われたのが最も古いページを入れ替えるものはどれか。

ア　FIFO　　イ　LFU　　ウ　LIFO　　エ　LRU

■基本　平成 13 年度（2001 年度）春期 午前・問 31

ページ置換えアルゴリズムにおける LRU 方式の説明として、適切なものはどれか。

ア　一番古くから存在するページを置き換える方式
イ　最後に参照されたページを置き換える方式
ウ　最後に参照されてからの経過時間が最も長いページを置き換える方式
エ　参照回数の最も少ないページを置き換える方式

■第２種　平成 12 年度（2000 年度）秋期 午前・問 28

仮想記憶管理におけるページ置換えに関する記述のうち、LRU 制御方式はどれか。

ア　各ページに参照フラグと変更フラグを付加して管理し、参照なし、変更なしのページを優先して置き換える。
イ　主記憶にあるすべてのページを同一の確率でランダムに選択し、置き換える。
ウ　最も長い間参照されていないページを置き換える。
エ　最も長い間主記憶にあったページを置き換える。

■第２種　平成 10 年度（1998 年度）秋期 午前・問 37

仮想記憶におけるページ置換えアルゴリズムの一つである LRU を説明した記述はどれか。

ア　あらかじめ設定されている優先度が最も低いページを追い出す。
イ　主記憶に存在している時間が最も長いページを追い出す。
ウ　主記憶に存在している時間が最も短いページを追い出す。
エ　最も長い間参照されていないページを追い出す

□第１種　平成 12 年度（2000 年度）午前・問 23

仮想記憶管理におけるページ置換えアルゴリズムとして、LRU 方式を採用する。参照かつ更新されるページ番号の順番が、2 → 3 → 5 → 8 → 2 → 3 → 6 → 2 → 3 → 5 → 1 → 6 で、実記憶のページ枠が 4 のとき、ページフォールトに伴って発生するページアウトは何回か。ここで、初期状態では、実記憶にはいずれのページも読み込まれていないものとする。

ア　3　　イ　4　　ウ　5　　エ　6

□第１種　平成 8 年度（1996 年度）午前・問 25

仮想記憶管理におけるページ置換えアルゴリズムとして、LRU 方式を採用する。参照かつ更新されるページ番号の順番が、2 → 3 → 5 → 8 → 2 → 3 → 6 → 2 → 3 → 5 → 1 → 6 で、主記憶のページ枠が 4 ページ分のとき、ページアウトは何回発生するか。ここで、初期状態では、主記憶ページには何も読み込まれていないものとする。

ア　2　　イ　3　　ウ　4　　エ　5　　オ　6

セグメンテーション方式

仮想記憶管理で使うマッピング方式には、ページング方式の他にセグメンテーション方式があります。

▶ セグメンテーション方式の特徴をページング方式に取り入れたのが、既に解説したセグメンテーションページング方式です。

セグメント方式では、プログラムやデータをモジュールの区切りや配列の区切りなどの論理的に関係のある構成単位に分割します。セグメント単位で仮想記憶領域に割り当て、セグメント番号と実アドレスの対応表であるセグメントテーブルを作成します。アドレス変換時に該当するセグメントが主記憶装置上にない場合は、仮想記憶装置から必要なセグメントを読み込んで、セグメントテーブルを書き換えます。実アドレスはセグメント番号とセグメント内の変位から求めます。**セグメントは可変長であるため、複雑な制御が必要となります**。

■基本　平成 14 年度（2002 年度）春期 午前・問 30

仮想記憶方式の一つに、プログラムの手続きやデータなど、論理的なひとまとまりを領域の単位として、仮想アドレス空間を分割して管理するものがある。この可変長の領域を示す用語はどれか。

ア　スロット　　イ　セクタ　　ウ　セグメント　　エ　フレーム

■第２種　平成８年度（1996 年度）秋期 午前・問 35

仮想記憶方式で用いられる記憶管理の手法又は機構を二つ選べ。

ア　オーバレイ　　イ　スプーリング　　ウ　動的アドレス変換
エ　動的リンキング　　オ　ページング

■第２種　平成７年度（1995 年度）秋期 午前・問 38

仮想記憶方式の記憶管理に関係の深いものを二つ選べ。

ア　キャッシュ　　イ　セグメント　　ウ　パイプライン
エ　ページ　　オ　リフレッシュ

■第２種　平成６年度（1994 年度）秋期 午前・問 27

オペレーティングシステムの機能とそれに関連する用語の組合せで、［　　］に入れるべき適切な字句はどれか。

ジョブ管理　－　JCL
［　　］　－　ページング
ファイル管理　－　ディレクトリ
入出力管理　－　バッファリング
運用管理　－　システムモニタリング

ア　記憶管理　　イ　システム管理　　ウ　装置管理
エ　通信管理　　オ　プロセス管理

■第２種　平成 10 年度（1998 年度）春期 午前・問 34

ページング方式に関する記述はどれか。

ア　実記憶空間と仮想記憶空間を、固定長の領域に区切り、対応づけて管理する方式
イ　主記憶装置の異なった領域で実行できるように、プログラムを再配置する方式
ウ　主記憶を、同時に並行して読み書き可能な複数の領域に分ける方式
エ　ファイル記憶媒体への読み書きをするとき、複数のレコードをまとめて行う方式

■基本　平成 13 年度（2001 年度）春期 午前・問 30

仮想記憶におけるセグメンテーション方式とページング方式に関する記述のうち、ページング方式の特徴はどれか。

ア　仮想アドレス空間の管理単位である領域の大きさを、実行時に動的に変更できる。
イ　実記憶領域の利用効率が高く、領域管理も容易である。
ウ　プログラムからみた論理的な単位でアクセス保護を行うことができる。
エ　プログラム実行中のモジュールの取込みや共有を容易に行うことができる。

■第２種　平成 11 年度（1999 年度）春期 午前・問 21

仮想記憶に関する記述のうち，正しいものはどれか。

ア　16 ビットのマシンでは，アドレス空間が小さいので仮想記憶が必要になるが、32 ビットのマシンでは、アドレス空間が大きいので仮想記憶は必要ない。
イ　仮想記憶を採用した場合，プログラム実行時のアドレスを前もって決めることができないので、動的リンキング機能を使ってプログラムを実行する。
ウ　実装された主記憶の容量を越えるアドレス空間の部分を補助記憶に割り付ける。
エ　プログラムの実行時にページ表（又はセグメント表）を使って、論理アドレスを物理アドレスに変換する。

■第２種　平成６年度（1994 年度）秋期 午前・問 16

仮想記憶に関する記述のうち、正しいものはどれか。

ア　仮想記憶空間は、主記憶から補助記憶へ連続したアドレスが割り付けられる。
イ　個々のプログラムに対しては、主記憶の容量を超えない範囲で仮想空間が与えられる。
ウ　主記憶に置かれていないプログラムは、キャッシュメモリ上で実行される。
エ　ページング方式は、仮想記憶を実現する一方法である。
オ　マルチタスクの実現には、仮想記憶が必要である。

□第１種　平成 11 年度（1999 年度）午前・問 96

仮想メモリに関する記述のうち、正しいものはどれか。

ア　実メモリに仮想ページが存在しないことをミスヒットと呼ぶ。
イ　ページサイズを大きくすると、スワップが発生しやすくなる。
ウ　ページサイズを小さくすると、スラッシングが発生しやすくなる。
エ　マルチタスキングでページフォールトを減らすには、プロセスの多重度をあげる。

□第１種　平成 12 年度（2000 年度）午前・問 28

主記憶装置をページに分割し、それぞれのページに R（読込み許可）、W（書込み許可）、E（実行許可）という属性を与えて、アクセスを制限しているシステムがある。このシステムにおいて、プログラムを実行する場合、コード領域、データ領域がロードされるページに付与される属性の組合せとして、最も適切なものはどれか。

	コード領域	データ領域
ア	ERW	ERW
イ	ER	RW
ウ	RW	RW
エ	R	W

■基本　平成 17 年度（2005 年度）秋期 午前・問 30

OS の記憶管理機能 a ～ c に対応する適切な用語の組合せはどれか。

機能	特　徴
a	あらかじめプログラムを幾つかの単位に分けて補助記憶に格納しておき、プログラムの指定に基づいて主記憶に読み込む。
b	主記憶とプログラムを固定長の単位に分割し、効率よく記憶管理する。これによって、少ない主記憶で大きなプログラムの実行を可能にする。
c	プログラムを一時的に停止させ、使用中の主記憶の内容を補助記憶に退避する。再開時には、退避した内容を主記憶に再ロードし、元の主記憶状態に戻す。

	a	b	c
ア	オーバレイ	ページング	スワッピング
イ	スワッピング	オーバレイ	ページング
ウ	スワッピング	ページング	オーバレイ
エ	ページング	オーバレイ	スワッピング

□ソフトウェア　平成 13 年度（2001 年度）午前・問 29

記憶管理機能の記述に関して、 a ～ c に対応する機能の正しい組合せはどれか。

機能	特　徴
a	あらかじめプログラムを幾つかの単位に分けて補助記憶装置に格納しておき、プログラムの指定に基づいて実記憶装置との間で出し入れをする。
b	主記憶とプログラムを固定長の単位に分割し、効率よく記憶管理する。このため、小さい主記憶装置で大きなプログラムの実行を可能にしている。
c	プログラムを一時的に停止させ、使用中の主記憶装置の内容を補助記憶装置に退避する。再開時には、退避した内容を主記憶装置に再ロードし、元の状態に戻す。

	a	b	c
ア	オーバレイ	ページング	スワッピング
イ	スワッピング	オーバレイ	ページング
ウ	スワッピング	ページング	オーバレイ
エ	ページング	オーバレイ	スワッピング

□第１種　平成 11 年度（1999 年度）午前・問 27

平成 13 年度 午前・問 29 と同じ

入出力管理

複数のジョブを同時に処理する多重プログラミングでは、処理装置に対して同時に入出力要求が発生することがあります。この問題を解決するのが、入出力管理の役割です。

制御方式

入出力の制御方式には、以下のような方式があります。

・プログラム制御方式

処理装置のレジスタを経由して、主記憶装置と入出力装置の間でデータ転送を行う方式のこと。CPUが入出力作業を担当することになるため、入出力中に、他の作業を行うと言ったことができません。

・DMA 制御方式

DMA は、direct memory access の略です。処理装置を介さずに、システムバスなどに接続されたデータ転送専用のハードウェアによって、主記憶装置と入出力装置の間で直接転送を行う方式です。

DMA コントローラが CPU に代わってデータ転送するため、高速転送が可能になるとともに、CPU はデータ転送中も別の処理が可能となります。

・チャネル制御方式

DMA 制御方式の一方式です。入出力装置の制御をチャネルが一括して行う方式であり、汎用コンピュータで広く採用されています。入出力の制御を専門に行う処理装置であるチャネルサブシステムが CPU とは独立・並行してデータ転送制御のためのプログラムを主記憶から自律的に読み出して入出力装置を制御するため、並行処理の度合いを高めることができます。

主記憶装置とチャネルとの間のデータ転送を専用のバスを介して行うため、一般的な DMA 方式で発生するバスの競合は発生しません。

チャネル制御方式では、実記憶と入力装置や出力装置の間に設置されたチャネルが、オペレーティングシステムの入出力監視プログラム中のチャネルプログラムの指示によって、入出力管理を行います。

チャネルには、セレクタチャネルとマルチプレクサチャネルがあります。

・セレクタチャネル selector channel（選択チャネル）

> **JIS X0011 01.25【セレクタチャネル】**
> **接続されている複数台の周辺装置のうち選択された 1 台と内部記憶装置との間のデータの転送を扱う機能単位。**

セレクタチャネルは、バーストモードで動作するチャネルです。バーストモードでは、入出力の開始から終了までの間、チャネルと入出力装置との物理的な接続が固定されます。磁気ディスク装置や磁気テープ装置などの入出力装置を制御するために使用されます。

・マルチプレクサチャネル multiplexer channel（多重チャネル）

> **JIS X0011 01.24【マルチプレクサチャネル】**
> **同時に動作する複数台の周辺装置と内部記憶装置との間のデータの転送を並行的に扱う機能単位。データ転送は、バイト単位又はブロック単位で行われる。**

マルチプレクサチャネルには、**バイトマルチプレクサチャネル**と**ブロックマルチプレクサチャネル**とがあります。

いずれのチャネルも処理装置と同じような機能をもち、処理装置から入出力の指令が出ると、入出力監視プログラムにあらかじめ用意されているチャネルプログラムがチャネルを動作させるようになっています。チャネルプログラムの指令によってチャネルが入出力管理を行うため、処理装置は入出力管理から解放され、別の処理を行うことができます。

スプーリング

周辺装置の動作を処理装置の動作と独立に並行して行わせるのがスプーリングです。プリンタなどの低速装置への出力データをいったん高速な磁気ディスクに格納しておき、その後に出力データを目的の装置に出力することによって、システムのスループットを高めます。

> **JIS X0010 04.01【スプーリング】**
> **周辺装置と計算機の処理装置との間でデータを転送するとき、処理の遅れを短縮するために補助記憶装置を緩衝記憶として用いること。〈注〉この用語は、語句“Simultaneous Peripheral Operation Online”に由来している。**

・バッファプール

プログラムの入出力と処理の並行動作を高めることによって性能向上を図ります。このために、主記憶上に入出力を行うための領域を多数用意し、複数のプログラムで共用します。

・デバイスドライバ

入出力装置に依存した処理を行い、装置の種類ごとに用意され、1 台又は複数台の装置を制御します。読出し、書込みなどの入出力要求が出されると、その装置を直接操作・管理します。

■基本　平成 14 年度（2002 年度）秋期 午前・問 32

低速の入出力装置や CPU などの処理効率向上を図るために、入出力データを一時的に磁気ディスク装置に蓄え、CPU や出力装置の空き時間に処理する機能はどれか。

ア　キャッシング　　イ　スプーリング
ウ　スラッシング　　エ　ページング

■第２種　平成 10 年度（1998 年度）春期 午前・問 36

スループットを高めるため、主記憶装置と低速の入出力装置とのデータ転送を、高速の補助記憶装置を介して行う方式はどれか。

ア　スプーリング　　イ　スワッピング　　ウ　ブロッキング　　エ　ポーリング

□第１種　平成 12 年度（2000 年度）春期 午前・問 27

システム全体のスループットを高めるため、主記憶装置と低速の入出力装置とのデータ転送を、高速の補助記憶装置を介して行う方式はどれか。

ア　スプーリング　　イ　スワッピング　　ウ　ブロッキング　　エ　ポーリング

■基本　平成 15 年度（2003 年度）・秋期 午前・問 33

スプーリングを行う目的はどれか。

ア　コンピュータシステムの運転経過の情報を記録する。
イ　物理レコードを意識することなく、論理レコード単位での処理を可能にする。
ウ　補助記憶装置を用いて、実記憶よりも大きな仮想記憶を提供する。
エ　補助記憶装置を用いて、低速の入出力を使用したときのシステムの処理効率を高める。

■基本　平成 18 年度（2006 年度）秋期 午前・問 30

スプーリングの説明として、適切なものはどれか。

ア　キーボードからの入力データを主記憶のキューにいったん保存しておく。
イ　システムに投入されたジョブの実行順序を、その特性や優先順位に応じて決定する。
ウ　通信データを直接通信相手の装置に送らず、あらかじめ登録しておいた代理の装置に送る。
エ　プリンタなどの低速出力装置へのデータをいったん高速な磁気ディスクに格納しておき、その後に目的の装置に出力する。

■基本　平成 16 年度（2004 年度）春期 午前・問 32

スプーリングの説明として、適切なものはどれか。

ア　キーボードからの入力データを主記憶のキューにいったん保存しておく。
イ　システムに投入されたジョブの実行順序を、その特性や優先順位に応じて決定する。
ウ　通信データを直接通信相手の装置に送らず、あらかじめ登録しておいた代理の装置に送る。
エ　プリンタなどの低速出力装置へのデータをいったん高速な磁気ディスクに格納しておき、その後に目的の装置に出力する。

■第２種　平成 11 年度（1995 年度）春期 午前・問 34

スプーリングの説明として，適切なものはどれか。

ア　キーボードなどからの入力データを主記憶上のキューにいったん保存しておく。
イ　通信データを直接通信相手の装置に送らず、あらかじめ登録しておいた代理の装置に送る。
ウ　二つ以上のプログラム（タスク、プロセス）に、時間で区切って CPU を割り当て、並行して実行する。
エ　プリンタなどの低速装置への出力データをいったん高速な磁気ディスクに格納しておき、その後に出力データを目的の装置に出力する。

■基本　平成 19 年度（2007 年度）・秋期 午前・問 27

スプーリング機能の説明として、適切なものはどれか。

ア　あるタスクを実行しているときに、入出力命令の実行によって CPU が遊休（アイドル）状態になると、ほかのタスクに CPU を割り当てる。
イ　実行中のプログラムを一時中断して、制御プログラムに制御を移す。
ウ　主記憶装置と低速の入出力装置との間のデータ転送を、補助記憶装置を介して行うことによって、システム全体の処理能力を高める。
エ　多数のバッファからなるバッファプールを用意し、主記憶にあるバッファをアクセスする確率を増すことによって、アクセス時間を短縮する。

■基本　平成 16 年度（2004 年度）・秋期 午前・問 31

スプーリング機能の説明として、適切なものはどれか。

ア　あるタスクを実行しているときに、入出力命令の実行によって CPU がアイドル状態になると、ほかのタスクに CPU を割り当てる。
イ　実行中のプログラムを一時中断して、制御プログラムに制御を移す。
ウ　主記憶装置と低速の入出力装置との間のデータ転送を、補助記憶装置を介して行うことによって、システム全体の処理能力を高める。
エ　多数のバッファからなるバッファプールを用意し、主記憶にあるバッファをアクセスする確率を増すことによって、アクセス時間を短縮する。

■第２種　平成 12 年度（2002 年度）・秋期 午前・問 34

スプーリング機能に関する記述として、適切なものはどれか。

ア　あるタスクを実行しているときに、入出力命令の実行によって CPU がアイドル状態になると、ほかのタスクに CPU を割り当てる。
イ　実行中のプログラムを一時中断して、制御プログラムに制御を移す。
ウ　主記憶装置と低速の入出力装置とのデータ転送を、補助記憶装置を介して行い、システム全体のスループットを高める。
エ　多数のバッファを含むバッファプールを用意して、主記憶にあるバッファをアクセスする確率を増やすことによって、アクセス時間を短縮する。

■第２種　平成 7 年度（1995 年度）春期 午前・問 32

スプーリングに関して、最も適切な記述はどれか。

ア　相手装置や通信ネットワークに関係なく、統一的な通信手順を提供する。
イ　外部記憶装置を用いて、主記憶装置より大きな仮想記憶を提供する。
ウ　コンピュータシステムの運転経過の情報を記録する。
エ　周辺装置の動作を処理装置の動作と独立に並行して行わせる。
オ　物理レコードを意識することなく、論理レコード単位での処理を可能とする。

■基本　平成 15 年度（2003 年度）春期 午前・問 33

スプーリング機能を使用してプリンタ出力を行うシステムがある。次の条件を満たすためには、スプーリングファイルの容量は少なくとも何 M バイト必要か。

〔条件〕
(1)　1 ジョブ当たりの印刷データは 2M バイトである。
(2)　スプーリングファイル上では、データは 50%に圧縮される。
(3)　1 時間当たり 100 ジョブを処理し、処理のばらつきは考慮しなくてよい。
(4)　最大 5 時間分の印刷データをスプーリングできる。

ア　100　　イ　250　　ウ　500　　エ　1,000

□第１種　平成 12 年度（2000 年度）春期 午前・問 19

入出力制御方式に関するａ～ｃの記述と用語の適切な組合せはどれか。

ａ　プロセッサのレジスタを経由して、主記憶装置と入出力装置の間でデータ転送を行う方式である。
ｂ　プロセッサを介さずに、システムバスなどに接続されたデータ転送専用のハードウェアによって、主記憶装置と入出力装置の間で直接転送を行う方式である。
ｃ　ｂの一方式であり、入出力専用のハードウェアがデータ転送制御のためのプログラムを主記憶から自律的に読み出して入出力装置を制御することによって、並行処理の度合いを高めることができる。

	a	b	c
ア	パイプライン制御方式	DMA 制御方式	チャネル制御方式
イ	パイプライン制御方式	チャネル制御方式	DMA 制御方式
ウ	プログラム制御方式	DMA 制御方式	チャネル制御方式
エ	プログラム制御方式	チャネル制御方式	DMA 制御方式

□ソフトウェア　平成 13 年度（2001 年度）春期 午前・問 27

入出力管理機能の記述に関して、ａ～ｃに対応する機能の正しい組合せはどれか。

機能	特　徴
a	入出力装置に依存した処理を行い、装置の種類ごとに用意され、1 台又は複数台の装置を制御する。読出し、書込みなどの入出力要求が出されると、その装置を直接操作・管理する。
b	ファイルのプリンタ出力やシリアル回線を介したファイル転送のように、それほど急を要さない入出力は、専用のプロセスに依頼して、入出力動作とプログラムの実行を並行して行う。
c	プログラムの入出力と処理の並行動作を高めることによって性能向上を図る。このために、主記憶上に入出力を行うための領域を多数用意し、複数のプログラムで共用する。

	a	b	c
ア	スプーリング	デバイスドライバ	バッファプール
イ	スプーリング	バッファプール	デバイスドライバ
ウ	デバイスドライバ	スプーリング	バッファプール
エ	デバイスドライバ	バッファプール	スプーリング

API

API（*application programming interface*）とは、アプリケーションプログラムがオペレーティングシステムの基本機能を使うためのコマンドの集まりであり、OSとアプリケーションとの仲介役を果たすものです。アプリケーションがAPIを介してハードウェアを操作するように規定されています。

JIS TR X0096 0802,1003の定義
APIは、アプリケーションプログラムインタフェースのことで、ある機能をアクセスするために使用する関数又はメソッドの集合とする。

■基本　平成20年度（2008年度）春期 午前・問29

OSにおけるAPI（Application Program Interface）の説明として、適切なものはどれか。

ア　アプリケーションがハードウェアを直接操作して、各種機能を実現するための仕組みである。
イ　アプリケーションから、OSが用意する各種機能を利用するための仕組みである。
ウ　複数のアプリケーション間でネットワークを介して通信する仕組みである。
エ　利用者の利便性を図るために、各アプリケーションのメニュー項目を統一する仕組みである。
※15秋34および14春34も同じ問題

■基本　平成16年度（2004年度）春期 午前・問33

パソコンのOSが提供する機能を利用するためのAPIに関する記述のうち、適切なものはどれか。

ア　APIで呼び出されるOSの処理モジュールは、あらかじめそれを利用するプログラムに静的にリンクしておく必要がある。
イ　OSのAPIが提供されない周辺機器は、ユーザプログラムから利用又は制御することはできない。
ウ　アーキテクチャの異なるCPU間でも、同じOSとそのAPIを使用することによって、プログラムの互換性を高め、移植時の工数を削減することが可能である。
エ　異なるOS間でもAPIは共通であり、APIだけを使用したプログラムであれば、再コンパイルだけでほかのOSへの移植が可能である。

コンソール画面制御

Windows の API を呼び出すと、コンソール画面に表示する文字に色を付けたり、画面上の任意の位置に文字を表示したりすることができます。

そのためのプログラムを以下に示します。

List 4　display.h

```
/*
    画面制御（エスケープシーケンス／Win32API）ヘッダ    "display.h"
*/

enum {  BLACK,              /* 黒 */
        BLUE,               /* 青 */
        RED,                /* 赤 */
        GREEN,              /* 緑 */
        MAGENTA,            /* 赤紫 */
        CYAN,               /* 水色 */
        YELLOW,             /* 黄色 */
        WHITE,              /* 白 */
        GRAY,               /* 灰色 */
        LIGHT_BLUE,         /* 明るい青 */
        LIGHT_RED,          /* 明るい赤 */
        LIGHT_GREEN,        /* 明るい緑 */
        LIGHT_MAGENTA,      /* 明るい紫 */
        LIGHT_CYAN,         /* 明るい水色 */
        LIGHT_YELLOW,       /* 明るい黄色 */
        BLIGHT_WHITE        /* 輝く白 */
};

/*--- 画面消去 ---*/
void cls(void);

/*--- カーソル位置を(__x, __y)に設定 ---*/
void locate(int __x, int __y);

/*--- 文字色を__fgに背景色を__bgに設定 ---*/
void colorx(int __fg, int __bg);

/*---  文字色を__colに設定 ---*/
void color(int __col);
```

List 5　display.c

```
/*
    画面制御（エスケープシーケンス／Win32API）
*/

#include  <stdio.h>
#include  "display.h"

#define ESCAPE_SEQUENCE 0       /* 1：エスケープシーケンス／0：Win32API */

#if (ESCAPE_SEQUENCE==0)
    #include  <windows.h>
#endif
```

```
/*--- 画面消去 ---*/
void cls(void)
{
#if (ESCAPE_SEQUENCE==1)
    printf("\x1B[2J");

#else
    HANDLE  hStdout = GetStdHandle(STD_OUTPUT_HANDLE);

    if (hStdout != INVALID_HANDLE_VALUE) {
        static COORD                coordScreen;
        DWORD                       dwCharsWritten;
        DWORD                       dwConsoleXY;
        CONSOLE_SCREEN_BUFFER_INFO  csbi;

        if (GetConsoleScreenBufferInfo(hStdout, &csbi) == FALSE)
            return;

        dwConsoleXY = csbi.dwSize.X * csbi.dwSize.Y;
        FillConsoleOutputCharacter(hStdout,
                    ' ', dwConsoleXY, coordScreen, &dwCharsWritten);
        FillConsoleOutputAttribute(hStdout,
                    csbi.wAttributes, dwConsoleXY, coordScreen,
&dwCharsWritten);
        locate(1, 1);
    }
#endif
}

/*--- カーソル位置を(x, y)に設定 ---*/
void locate(int x, int y)
{
#if (ESCAPE_SEQUENCE==1)
    printf("\x1B[%d;%dH", y, x);

#else
    HANDLE  hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD   coord;

    if (hStdout != INVALID_HANDLE_VALUE) {
        coord.X = x - 1;
        coord.Y = y - 1;
        SetConsoleCursorPosition(hStdout, coord);
    }
#endif
}

/*--- 表示色をfgに背景色をbgに設定 ---*/
void colorx(int fg, int bg)
{
#if (ESCAPE_SEQUENCE==1)
    int  col[] = {30, 34, 31, 32, 35, 36, 33, 37};

    printf("\x1B[0;");
    if (fg > WHITE) printf("1;");       /* 高輝度 */
    printf("%d;%dm", col[fg % 8], col[bg % 8] + 10);

#else
    int  col[] = {0, 1, 4, 2, 5, 3, 6, 7, 8, 9, 12, 10, 13, 11, 14, 15};
    HANDLE  hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
    WORD    attr;
```

```
    if (hStdout == INVALID_HANDLE_VALUE)
        return;

    attr = (col[bg % 16] << 4) | col[fg % 16];

    SetConsoleTextAttribute(hStdout, attr);
#endif
}

/*--- 文字色をcolに設定 ---*/
void color(int col)
{
    colorx(col, BLACK);     /* 文字色はcolで背景色は黒 */
}
```

ライブラリの使い方

重要 !!　必ず覚えましょう。

・"display.h" をインクルードする。
・"display.c" をプロジェクトに登録する。

各関数の概要は、以下のとおりです。

void cls(void);

画面を消去します。

void locate(int x, int y);

カーソル位置を (x, y) に設定します。
座標は、画面の左上隅が (1, 1) です。

void colorx(int fg, int bg);

文字色を fg に、背景色を bg に設定する。

void color(int col);

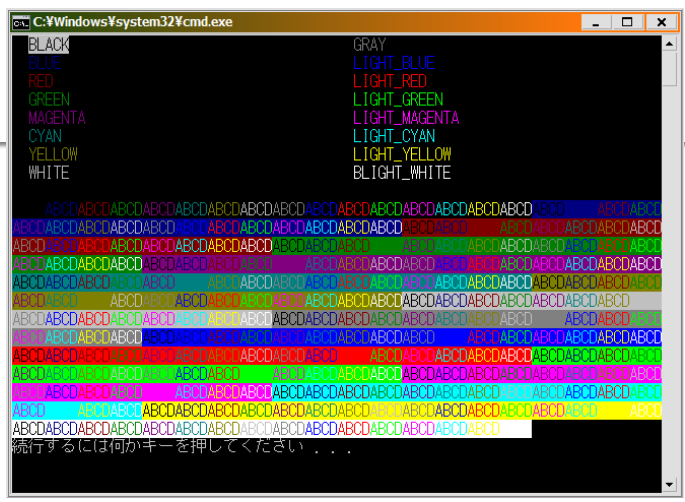
文字色を col に設定する（背景色は黒になります）。

ライブラリの利用例

ライブラリを利用したプログラム例を示します。

List 6 disptest.c

```
/*
    画面制御テストプログラム
*/

#include <stdio.h>
#include "display.h"

int main(void)
{
    int i, j;

    cls();      /* 画面消去 */

    locate( 3, 1);  colorx(BLACK, WHITE);   printf("BLACK");
    locate( 3, 2);  color(BLUE);            printf("BLUE");
    locate( 3, 3);  color(RED);             printf("RED");
    locate( 3, 4);  color(GREEN);           printf("GREEN");
    locate( 3, 5);  color(MAGENTA);         printf("MAGENTA");
    locate( 3, 6);  color(CYAN);            printf("CYAN");
    locate( 3, 7);  color(YELLOW);          printf("YELLOW");
    locate( 3, 8);  color(WHITE);           printf("WHITE");
    locate(43, 1);  color(GRAY);            printf("GRAY");
    locate(43, 2);  color(LIGHT_BLUE);      printf("LIGHT_BLUE");
    locate(43, 3);  color(LIGHT_RED);       printf("LIGHT_RED");
    locate(43, 4);  color(LIGHT_GREEN);     printf("LIGHT_GREEN");
    locate(43, 5);  color(LIGHT_MAGENTA);   printf("LIGHT_MAGENTA");
    locate(43, 6);  color(LIGHT_CYAN);      printf("LIGHT_CYAN");
    locate(43, 7);  color(LIGHT_YELLOW);    printf("LIGHT_YELLOW");
    locate(43, 8);  color(BLIGHT_WHITE);    printf("BLIGHT_WHITE");

    printf("\n\n");
    for (i = 0; i < 16; i++) {
        for (j = 0; j < 16; j++) {
            colorx(j, i);
            printf("ABCD");
        }
    }
    color(WHITE);
    printf("\n");

    return (0);
}
```



参考文献

1) Peter van der Linden、梅原 系（翻訳）『エキスパート C プログラミング―知られざる C の深層』，アスキー，1996

2) 財団法人 日本情報処理開発協会中央情報教育研究所 監修『第二種共通テキスト② ソフトウェア』，コンピュータ・エージ，1998

3) 情報処理研究会『平成 10 年度 第 1 種情報処理技術者試験解答集』，電気書院，1997

4) 『2001 年度版ソフトウェア開発技術者完全解答』，オーム社，2001

5) 情報処理研究会『平成 9 年度春期版 第 2 種情報処理技術者試模範験解答集』，電気書院，1997

6) 河村知信、金子崇『’99 年度版 第 1 種情報処理技術者合格完全対策』，経林書房，1999

7) 河村知信『2000 年度版 第 1 種情報処理技術者第 2 種題＆分析』，経林書房，1999

8) 藤本喜弘、河村知信『’98 年度春・秋試験対応版 第二種情報処理技術者試験第 2 種題＆分析』，経林書房，1997

9) 柴田望洋『新版明解 C 言語実践編』，ソフトバンクパブリッシング，2004