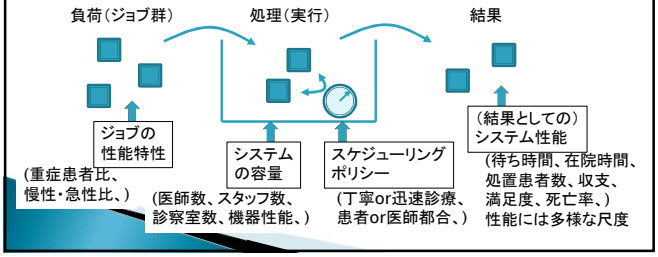
第15章 OSと性能

花田 英輔
（このPowerPointは渡辺名誉教授作成のものを花田が一部改編した）

1

性能の要素(教科書15.1)

- 処理する物のサイズ：プロセッサ速度、メモリ容量等
- 処理される物のサイズ：プログラムの命令数等
- OSにおけるスケジューリング方針
- 実際の処理量：単位時間当たり処理量、応答時間等



2

システムの容量

- 処理する物のサイズ
  - プロセッサ性能：単位当たりの命令実行数
  - メモリ性能：容量
  - ディスク性能：容量、読み書き速度
  - 入出力性能：単位時間当たりのデータ転送量

3

システムの性能指標

- プロセッサ性能：単位当たりの命令実行数（命令数/時間
  - 単位：MIPS(Million Instructions Per Seconds)等
  - クロック周波数(単位：GHz(Giga Hertz)等)が関係
  - 他にもキャッシュサイズ、コア数、命令セット等が関係
- メモリ性能：容量：
  - 単位：MB(Mega Bytes)、GB(Giga Bytes)等
  - 読み書き速度も関係
- ディスク性能：容量、読み書き速度
  - 単位：GB(Giga Bytes)、TB(Tera Bytes)等
  - シーク時間（ヘッド移動）、サーチ時間（回転待ち）が関係
- 入出力性能：単位時間当たりのデータ転送量
  - 単位：MB/秒等

命令セット
RISC（Reduced Instruction Set Computer）
命令種類を減らし回路を単純化・高速化
CISC（Complex ..）：RISCに対する語、旧来型
＊結局、ソフト互換性からCISCが勝利

4

単位表記

K(Kilo)	$=10^3=1,000$
M(Mega)	$=10^6=1,000,000$
G(Giga)	$=10^9=1,000,000,000$
T(Tera)	$=10^{12}=1,000,000,000,000$
P(Peta)	$=10^{15}=1,000,000,000,000,000$
E(Exa)	$=10^{18}=1,000,000,000,000,000,000$
Z(Zetta)	$=10^{21}=1,000,000,000,000,000,000,000$
Y(Yotta)	$=10^{24}=1,000,000,000,000,000,000,000,000$

m(mili)	$=10^{-3}=1/1,000$
μ(micro)	$=10^{-6}=1/1,000,000$
n(nano)	$=10^{-9}=1/1,000,000,000$
p(pico)	$=10^{-12}=1/1,000,000,000,000$
f(femto)	$=10^{-15}=1/1,000,000,000,000,000$
a(atto)	$=10^{-18}=1/1,000,000,000,000,000,000$
z(zepto)	$=10^{-21}=1/1,000,000,000,000,000,000,000$
y(yocto)	$=10^{-24}=1/1,000,000,000,000,000,000,000,000$

ビット・バイト表現では、2^{10}=1,024をK, 2^{20}=1,048,576をM等との表記も使用
混乱を避けるため、MiB(mebibyte, mega binary byte) 等と表記することも

5

ジョブのサイズ

- 処理される物のサイズ
  - ジョブ
    - 利用者が実行するひとまとまりのプログラム（特にバッチ処理）
    - 性能に関係する様々な特性＝実行命令数、所要メモリ量、入出力回数、入出力データ量
- 比較には標準的なジョブ群を定義する必要
  - ベンチマークジョブ

6

ベンチマーク

- 測量において利用する水準点
  ⇒ハードやソフトの性能を測定するための指標
  - ベンチマークテスト
    - プログラムを走らせて性能を比較する
    - 実際に業務で使うプログラムまたは標準的な処理のプログラム
  - 有名なベンチマーク用プログラム
    - Dhrystone – 整数演算性能
    - Whetstone – 浮動小数点演算性能
    - SPEC – 総合性能、マルチプロセッサ性能
    - TPC-C – トランザクション性能
    - LINPACK – 浮動小数点演算性能

7

スケジューリングポリシー

- OSにおけるスケジュール方針
  - 方針によって各ジョブの待たされ方が変化
    ⇒応答時間が変化
    方式例)
    - FCFS(First Come First Serve)
    - 優先度方式
    - ラウンドロビン

8

スケジューリングポリシー

- FCFS(先に来たものを先に)
  - 長いジョブが挟まると後続ジョブが長時間待ち
- 優先度方式(優先度の値の高いものを先に)
  - 優先度の高いジョブは高速、低いジョブは低速、実行不可も
- ラウンドロビン(短時間ごとに順次切替えて実行)
  - 固有の実行時間とほぼ比例した応答時間
    =平等配分できる可能、頻繁に切替えのためオーバーヘッドが大

オーバーヘッド(overhead)=間接費
何らかの処理を進める際に、間接的・付加的に必要となる処理とそれにより発生する負荷の大きさ(教科書15.6)

9

システムの性能(教科書15.2)

- 様々な視点
  - システム全体、各資源、各ジョブ、各利用者
- 様々な評価基準
  - スループット:システムの処理量
    - 単位時間当たりの処理件数(件数/時間)
      (前提となるジョブを定義の必要(例:1件の問合せ処理))
  - 資源の利用率
    - 順次使用資源(例:プリンタ)
      ⇒使用している時間長/全時間長
    - 空間使用資源(例:ディスク)
      ⇒使用している領域サイズ/全領域サイズ

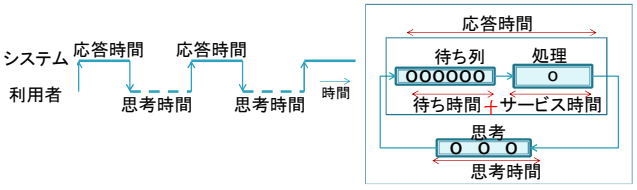
10

システムの性能

- ジョブの経過時間
  - プロセッサ実行時間(時間/ジョブ)
  - 入出力処理時間(時間/ジョブ)
  - 資源の空き待ち時間(時間/ジョブ)
    - プロセッサ待ち、メモリ待ち、入出力待ち、排他的資源待ち
- 端末の応答時間(Response Time)
  - 端末から要求してから、その応答が返るまでの時間
    - 比較には処理内容を定義する必要、他利用者の存在影響
  - 分散が大きいと不快

11

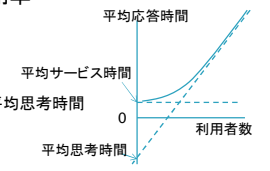
端末の応答時間(教科書15.4)



$$\text{平均応答時間} = \frac{\text{平均サービス時間} \times \text{利用者数}}{\text{サーバ利用率}} - \text{平均思考時間}$$

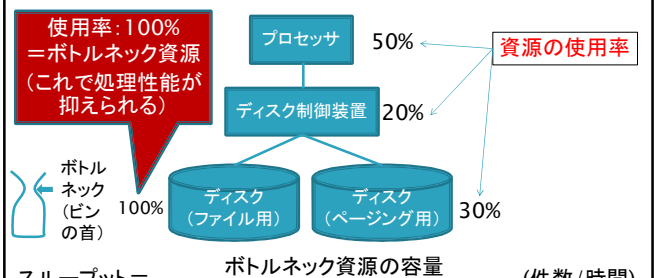
利用者数1では待ち時間なし
平均応答時間=平均サービス時間

利用者数大では利用率100%
平均応答時間=平均サービス時間×利用者数 – 平均思考時間



12

資源の使用率とスループット(教科書15.5)



$$スループット = \frac{ボトルネック資源の容量}{ジョブ当たりの、この資源の平均使用量} - (件数/時間)$$

例:100MBディスクを1ジョブが10MB使用 ⇒一度に最大10ジョブが使用可能

ボトルネック以外を増強しても効果なし
例:プロセッサを倍の性能にしてもスループットは同じ

13

第16章 OSと標準化

14

文字コード(教科書16.1)

- 当初は英語しかなかった
  - アルファベット＋若干の記号⇒文字は100種もあれば十分
  - 1文字を、7ビット　または　8ビット　で表現
- ASCII:7ビット表現⇒小型機用から普及、現在一般的
  - American standard code for information interchange
  - 49(0x31)=‘1’、50(0x32)=‘2’、65(0x41)=‘A’、
- EBCDIC:8ビット表現⇒汎用機用から普及
  - Extended binary coded decimal interchange code

10進数、	16進数、	8進数表現
49	0x49	049
そのまま	0xを頭に	0を頭に

15

JISコードの1バイトコード(制御コード、ASCII文字)のエリアマップ

ASCIIコード

16進数 2進数	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	a 1010	b 1011	c 1100	d 1101	e 1110	f 1111
0 0000	0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL	8 BS	9 HT	10 LF	11 VT	12 FF	13 CR	14 SO	15 SI
1 0001	16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB	24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
2 0010	32 SP	33 !	34 "	35 #	36 $	37 %	38 &	39 '	40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
3 0011	48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
4 0100	64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
5 0101	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
6 0110	96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
7 0111	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

上位4ビット　下位4ビット

‘R’=0101 0010

16

日本語のサポート

- 英文字定義(ほぼASCII):JISX0201ラテン文字
  - ASCIIと2文字違い
  - バックスラッシュ[\]⇒円マーク[¥]に
  - チルダ[~]⇒オーバーライン[‾]に
- 片仮名定義:JISX0201片仮名
  - 49(0x31)=‘ｱ’、50(0x32)=‘ｲ’、

17

JISコードの1バイトコード(半角カタカナ)のエリアマップ

半角カタカナ

16進数 2進数	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	a 1010	b 1011	c 1100	d 1101	e 1110	f 1111
0 0000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 0001	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2 0010	32	33 。	34 「	35 」	36 、	37 ・	38 ヲ	39 ァ	40 ィ	41 ゥ	42 ェ	43 ォ	44 ャ	45 ュ	46 ョ	47 ッ
3 0011	48 ー	49 ア	50 イ	51 ウ	52 エ	53 オ	54 カ	55 キ	56 ク	57 ケ	58 コ	59 サ	60 シ	61 ス	62 セ	63 ソ
4 0100	64 タ	65 チ	66 ツ	67 テ	68 ト	69 ナ	70 ニ	71 ヌ	72 ネ	73 ノ	74 ハ	75 ヒ	76 フ	77 ヘ	78 ホ	79 マ
5 0101	80 ミ	81 ム	82 メ	83 モ	84 ヤ	85 ユ	86 ヨ	87 ラ	88 リ	89 ル	90 レ	91 ロ	92 ワ	93 ン	94 ゛	95 ゜
6 0110	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7 0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

上位4ビット　下位4ビット

‘メ’=0101 0010

18

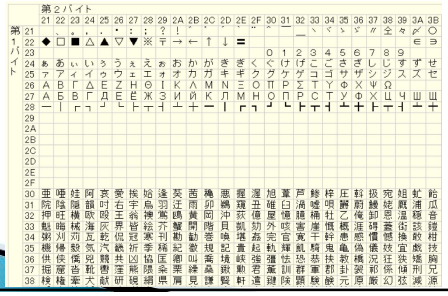
文字符号化方式

- 保持・送信等には工夫が必要
  - ‘R’=0101 0010
  - ‘メ’=0101 0010
- 符号化
  - 文字に対応付けた非負整数値
    ⇒実際にコンピュータが利用できるデータ列に変換
  - 方法A:エスケープシーケンスで、続くコードの種類を指示
    - ESC $ B 英数字コード列 ESC (I 半角カタカナコード列
    - 主に通信伝送にて使用
  - 方法B:半角カタカナは最上位ビットを1に英数字コードは0に
    - ‘R’=0101 0010、‘メ’=1101 0010
    - 主にメモリ保持にて使用

19

漢字コード

- 英数字、カタカナは7ビット(0~127)の値に対応
- ひらがな、漢字も⇒7ビット(0~127) ×2の値に対応
- JIS漢字コード JIS X 0208
- ‘亜’=30 21 (16進数)



20

漢字を含む文字の符号化法

- これも1バイト文字とコードが重なる
  - ‘亜’=0x30 0x21、 ‘0’ =0x30 、‘!’ =0x21
- JIS符号化
  - エスケープシーケンスで、続くコードの種類を指示
  - ネットワークで標準的
- シフトJIS符号化
  - 最上位ビットが0は英数字、1は半角カタカナ
  - 最上位ビットが1で未使用部⇒漢字第1バイトで使用
  - 従来のPCで標準的
- EUC符号化
  - 半角カタカナをあきらめ、最上位ビット1は漢字とする
  - 従来のUNIXで標準的

21

JIS(日本産業規格)漢字コード

- JIS X0208
  - 7ビット及び8ビットの2バイト情報交換用符号化漢字集合
  - 第1水準漢字集合(一般日本語表記用漢字)及び第2水準漢字集合(地名・人名を含む個別分野用漢字)の漢字6,355字を収録

  ⇩ その他の漢字も表現
- JIS X0213
  - 7ビット及び8ビットの2バイト情報交換用符号化拡張漢字集合
  - 第1・2水準の漢字に加え、第1・2水準では足りない文字を中心に選定した第3水準漢字集合及び第4水準漢字集合の漢字3,695字を収録(総計10,050字の漢字)

  ⇩ 世界中の文字も表現
- JIS X0221
  - 国際符号化文字集合(UCS)、国際規格ISO/IEC 10646の翻訳規格

22

国際符号化文字集合(UCS)

- UCS=全世界の文字を32bits(4Bytes)に包含した体系
- 符号化方法
  - UTF-32:そのまま4バイトで保持=英語中心のファイルでは無駄
  - UTF-8:1~6バイトで保持、ASCII=1バイト、漢字=3バイト、ほとんど使わない文字ほど多バイト
    - ASCII文字と互換性を持たせるためASCIIと同じ部分は1バイト、その他の部分を2-6バイトで符号化
  - UTF-16:英語、日本語相当分を2バイトで保持、Unicode同等、UCSコードの全てをカバーしない
- 普及拡大中、Windows、UNIX、Java、Web等で標準的に
  - UTF-8で符号化が一般的

23

文字化け

- 文字の符号化と復号でコードが相違すると文字化け

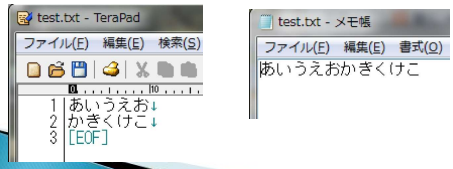
- エディタ・ブラウザによっては自動判定やマニュアル指定も可能

24

改行コード

- 改行を表すコードもOSにより相違
- システム間移動で改行がおかしくなることも

OS	改行コード	2進表現	16進表現
UNIX、MacOS X	LF	00001010	0x0A
Windows	CR+LF	00001101 00001010	0x0D 0x0A
Macintosh (Ver.9以前)	CR	00001101	0x0D



25

OSの国際化と地域化(教科書16.2)

- 国際化:internationalization (i18n)
  - ソフトウェアに技術的な変更を加えることなく多様な言語や地域に適合できるようにする
  - ロケール＝利用者の言語や文化に関する実行環境定義 setLocale(L)によって設定、同関数(例date())で異なる表示
    - 文字、文字コード
    - 日付表示(2001/05/02、02/05/2001、May 2, 2001.)
    - 数値表示 (1,000,000、100,0000、1.000.000.)
- 地域化:Localization (L10n)
  - 地域固有の構成部品や翻訳テキストを追加することによって、ソフトウェアを特定の地域や言語に適合させること
    - かな漢字変換、、

26

OSの仕様の標準化(教科書16.3)

- API標準化
  ⇒応用プログラム開発者にメリット
- ユーザインタフェース標準化
  ⇒利用者にメリット
- 標準化ドキュメンテーション
  ⇒OS開発者にメリット
- オープンシステム
  - 標準化され、特定の所有者ではないオープンな仕様に基づくシステム

27

UNIX系OSの仕様の標準化

- UNIX系のオープンシステム標準
  - 国際標準: ISO POSIX
  - The Open Group: 単一UNIX仕様
- 「UNIX」
  - 当初はAT&Tベル研で開発されたOS実体
  - 現状では仕様の名前、許諾を得たOSのみUNIXを名乗れる
    - AIX,HP-UX,Solaris,MacOSXなど
  - 許諾なしのOSは、UNIXライク(UNIX系)とされる
    - Linux,FreeBSD,NetBSD,OpenBSDなど

28

今回の課題

1. マルチプログラミング環境において、プロセッサ使用率が50%、入出力装置使用率が100%であり、システム内のプロセス数は十分に多いとする。以下の()内を埋めよ。
   1) プロセッサ性能を2倍にしたらスループットは( )倍になる。
   2) 入出力性能を2倍にしたらスループットは( )倍になる。
2. この講義全体の感想を書いてください

- 今回のファイル名は“学籍番号-OS15.docx”
  (例:23238000-OS15.docx)としてください
- 締切:2月7日(金) 18:00 (遅れた場合は減点)

29