平成14年度オペレーティングシステム試験

2003年2月10日

問題は3問，5ページある．

1

次の文章を読んで以下の問いに答えよ．

桂子: 博士，最近D-inkとかいう，消しゴムで消せるボールペンなんていうものが売られていますね

博士: そうじゃな，あのコマーシャルおもろいな.『うわー，消すな〜，ワシの理論を消すな〜』なんてな．そこへ行くと，わしの研究成果は全部このちゃーんとしたオペレーティングシステムの管理するハードディスクの中に入ってるから，オマエさんのような不届き者がいくら消そうと試みても消されることはないんじゃ

桂子: 本当ですか? 私には信じられません．だって，オペレーティングシステムといえども単なるソフトウェア．消されることはないなんて信じられません

博士: わかっとらんな．君は大学で何を学んどるのじゃ．ためしにこのコンピュータにログインして，わしのファイルを消してみい

〈桂子, ログインして，ファイルを消すコマンドを実行してみる．もちろんそれは失敗する．〉

桂子: Permission Deniedだって... 本当ですね

博士: はっはっは．そのファイルを消せるのはワシだけじゃ.

〈大作登場〉

大作: そんなことでは，桂子さんはだまされても，僕はだまされませんよ，先生

博士: だますとは人聞きが悪いな．でもそうやって，素朴な疑問をぶつけるのは良いことじゃ．今，3日後に〆切の論文を書いていて忙しいのじゃが，今日は特別じゃ．いうてみい

大作: 桂子さんのアクセスが拒否されたのは，オペレーティングシステムが管理するファイルの属性の中に『桂子さんはそのファイルを消せない』ということが書かれていて，オペレーティングシステムのコードのどこかで，それを検査するコードが書かれていたからです．ハードディスクの本体に，誰がファイルを消そうとしたかを検査する機能があるわけではありません

博士: うむ，オマエさんは少しはわかっておるようじゃな

大作: 僕は昔，ハードディスクのデバイスドライバ (注: オペレーティングシステムに組み込まれて，ハードディスクを直接アクセスするプログラム) を書いたことがあります．だからその時のプログラムを使えば，オペレーティングシステムのコードなんて使わなくても，ハードディスクの読み書きができます．つまり，そのオペレーティングシステムの検査コードを通らずにハードディスクが読み書きができるはずです

博士: ははは，やってみい

〈 2 日後，大作はそのときのコードを持って博士の部屋を訪れる．〉

大作: 博士，持ってきました．さーて，このプログラムをコンパイルして，と．さあ，実行するぞ．ハードディスクをめちゃくちゃにしてやる．

博士: ははは，やってみい

〈 大作, 実行する 〉

大作: あれぇ〜 (1) だって

博士: 当たり前じゃ，君のような不届き者が (2)ハードディスクなどに直接アクセスする命令を発行してもできないような仕組み が今時のコンピュータには必ず備わっておるのじゃ

桂子: ふーん，よくできているんですね

博士: なんじゃ，桂子君，今日もおったのか

桂子: これで，ハードディスクに直接アクセスしようとしてもできないということはわかりました．でも，まだ質問があります

博士: な，何かな．ろ，論文の〆切が...

桂子: 私たちは普段オペレーティングシステムが提供する API を呼び出して，自分のファイルに書いたり，ファイルを消したりしていますよね

博士: そうじゃな

桂子: ということは，私たちの書いたプログラムから，オペレーティングシステムの内部にジャンプする命令がどこかで実行されているはずですよね

博士: いかにもじゃ

桂子: そのジャンプ命令のあて先は，私たちの書いたプログラムに書いてあるわけだから，オペレーティングシステムのどこに，ファイルを消す命令が書いてあるかを突き止めれば，その場所に普通のジャンプ命令でジャンプしちゃえばいいんじゃないかしら?

博士: そんな悪知恵は通用せんのじゃ．(3)オペレーティングシステム内部に，普通のジャンプ命令を使って勝手にジャンプすることはできないように，守られておるのじゃ．それだけじゃなく，オペレーティングシステムがわしのために読み出して，メモリにためてあるデータを，オマエさんが勝手に読むことも，書くこともできないようになっておる のじゃ.

桂子: ふうん，じゃ，勝手な番地にジャンプする代わりに，オペレーティングシステムのコードを「ちゃんと」呼び出すための何か別の仕組みがあるのね?

博士: そうじゃ，それを (4) 命令というのじゃ．その場合のジャンプ先は，コンピュータのメモリ中のある領域に書かれており，(5) と呼ばれているのじゃ．それは通常オペレーティングシステムが起動するときにオペレーティングシステムによって設定されるのじゃ

桂子: ふーん，じゃ，その [(5)] を勝手に書き換えたら? あ，そうか，それもさっきの『オペーレーティングシステム内部に，普通のジャンプ命令を使って勝手にジャンプすることはできないように』するのと同じ仕組みで禁止するわけですね.

博士: そうじゃ，そうじゃ，やっとわかってきたようじゃの．ワシも論文の〆切前にオマエさんたちにつきあった甲斐があったというものじゃ

桂子: はい，オペレーティングシステムって，結構巧妙に自分自身が破壊されないように作られてるんですね．勉強になりました．...あれ，ここに書いてある『管理者用パスワード x84(a!) 』って何かしら？エエっとこれを打ち込むと... あ，管理者になれたみたい．これで博士のファイルを消してみよーっト．あー，簡単に消えちゃった

博士: う，うわー，消すな〜，ワシの論文を消すな〜

桂子: いくらオペレーティングシステムがちゃんとしていても，博士がこんなところに管理者用パスワードをおきっぱなしにしていては何にもなりませんね，博士

博士: い，いかにもじゃ...オマエさんたちはくれぐれも気をつけるのじゃ...

(1) [(1)] で大作が観測した現象はどのような現象と推察されるか? 下線部 (2) とのつながりを考慮して答えよ．答え方としては，ありそうなエラーメッセージを答えても良いし，エラーの内容を説明してもよい.

(2) 下線部 (2) でいう，仕組みについて簡潔に説明せよ.

(3) 下線部 (3) の仕組みについて簡潔に説明せよ.

(4) [(4)] に当てはまる適当な言葉を答えよ.

(5) [(5)] に当てはまる適当な言葉を答えよ.

2

以下のような処理を行うスレッドをいくつか同時に走らせることを考える．ただし，PAGESIZE は 1 ページのサイズ (バイト数) である．

```
thread() {
    char * A =OS から n ページ分のメモリを確保;
    while (1) {
      task(A);
    }
}
```

ここで，task は以下のような処理である．

```
task(A) {
    for (i = 0; i < n; i++) {
      x = A[i * PAGESIZE];                    — (*)
      compute(x);
    }
}
```

つまり，スレッドは task と称する処理を繰り返し (永遠に) 行い，task は配列 A の要素を先頭から，1 ページごとに 1 要素，順に読み出しては，各要素ごとにある計算 (compute) を行う．また，以下を仮定する．

- compute の中では I/O も行わず，ほとんどメモリもつかわない，CPU のみを消費する処理である．特に compute 中では配列 A にもアクセスしない．すなわち，各スレッドがアクセスするメモリ場所の集合は，上記の $(*)$ 部でアクセスされる配列 A の要素がほとんどであり，他は無視できるものとする．
- 各 compute は，1 台の，他に何もしていない CPU によって a [ms] かかる程度の処理である．
- OS は LRU (Least Recently Used) ページ置換を行う．
- OS のページフォルト処理，つまり，物理メモリ中の 1 ページを除去して，替わりの 1 ページをディスクから読み込んでくるまでの処理時間は b [ms] である．複数のページを並行して入れ替えることができ，ディスクのバンド幅は無限とする．
- 本問題のコンピュータは P ページ分の物理メモリを持つ．このコンピュータ上で，上で示した以外のスレッドが使うメモリはほとんど無視できるものとする．

全スレッドが実行を開始してから十分長い時間が経過した後，全スレッド合計で，1 ms あたりに終了することのできる task の個数をシステムのスループットと呼ぶことにする．

以下の問いに答えよ．すべての問いについて，計算の根拠も簡単に述べよ．

(1) スレッドを 1 つだけ走らせたとする．n を色々変えたときのシステムのスループットを求めて，大体の様子を横軸を n，縦軸をスループットとしたグラフに書け．

(2) n を固定して，スレッド数 t を色々に変える．横軸を t, 縦軸をスループットとしたグラフを書け．

注: a, b, n の大きさ (の比) によって，グラフの形が変わるかもしれない．よく考えて，必要ならば場合分けせよ．

3

オペレーティングシステムのスレッド (プロセス) スケジューラは，以下の目標を達成するべくスレッドをスケジューリングしている．

- CPU を無駄にしない
- スレッド間になるべく公平に CPU 時間を割り当てる
- 多数のスレッド (プロセス) が実行していても，対話的なプログラム (エディタなど) の応答時間が損なわれない

このための仕組みについて，具体的に 説明せよ．説明としては,「対話的プログラムの優先度を上げる」などという抽象的な説明では足りない．具体的に，スケジューリングに関係するさまざまな事象と，事象が発生した際にオペレーティングシステムがどのような動作を行うかを説明せよ．そして，なぜこれで上記の目的がよく達成されるのかを説明せよ．

注: 授業ではそのような方式の一例について具体的に説明したが，それと違う答えも十分ありうる．単に暗記力を問うているのではない．

問題は以上である