システムソフトウェアと
オペレーティング・システム 90－22
（２００２．６．２８）

体験学習モデルによるOSの学習システムの実現

米田 孝文[*1]　　早川 栄一[*2]　　高橋 延匡[*2]

[*1]拓殖大学大学院 工学研究科 電子情報工学専攻
[*2]拓殖大学 工学部 情報工学科

[*1]yoneda@os.cs.takushoku-u.ac.jp, [*2]hayakawa@cs.takushoku-u.ac.jp

本稿では、OS の資源管理方式を可視化した、体験学習モデルによる OS の学習システムの設計について述べる。OS を学ぶ学生を対象として、資源管理の振舞いを理解してもらうために、ユーザに OS の振舞いに沿って資源管理を行わせて、その仕組みを学習できる仮想環境を提供する。これによって、ユーザは、実ハードウェアの制約を受けずに、資源管理の全体構成や、各アルゴリズムなどを理解することが可能になる。プロトタイプとして、ファイルシステムを中心に実現を行った。

Implementation of the Learning System for Operating System modeled on learning by experienced

Takafumi Yoneda[*1]　　Eiichi Hayakawa[*2]　　Nobumasa Takahashi[*2]

[*1]Graduate School of Engineering, Takushoku University
[*2]Takushoku University

[*1]yoneda@os.cs.takushoku-u.ac.jp, [*2]hayakawa@cs.takushoku-u.ac.jp

This paper describes the design of an Operating System (OS) learning system that visualizes its resource management based on experience. Understanding the behavior of resource management in operating system course, this system supports a virtual environment that makes a user manage hardware resources as a real operating system. Learners can understand an overall structure or each algorithm in operating system without the limitation of the hardware. Now prototype system is implemented that mainly presents file system facilities.

1　はじめに

現在のユーザは、OS の資源管理を学習するときに戸惑う問題を抱え込んでいる。例えばユーザが、それを学習する場合、一つ目に、教科書、参考書を読んだり、人に教わるなどの概念から学習する方法がある。二つ目には、実際の OS のソースコードを読んで、実装レベルから学習する方法がある。しかし、このどちらも、OS の全体が見えないまま学習が進むため、OS が扱う多くのデータ構造の関係が見えづらい。だからユーザは、戸惑いを起こす。つまり OS の資源管理を学習することにおいて、多くの概念の仕組みと、その関係を見せることは、ユーザの戸惑いを解消するためにも効果的である。

本報告は、体験学習モデルによる OS の学習システムの実現について述べる。

2　本研究の目的

本研究の目的は、ユーザが OS の役割を担い、OS の振る舞いを真似ることによって、資源管理方式を体験、学習できる環境を実現することである。これによってユーザは、資源管理方式を自分の目と手で体験することができるようになる。そしてユーザは、OS の資源管理の仕組みを深く理解し、1 で挙げた問題を解決できると考えた。本来、OS のない時代、その管理は人間が行っていた。そのため、本研究における、人間が OS の役割を担うという環境の実現は妥当である。

3　設計方針

システムの設計方針について述べる。

(1) ユーザを OS になったつもりにさせる

ユーザが OS の立場を担い、仮想 OS として資源管理方式を行うことによって、その学習効果を高めることを目指す。そのため、実際のハードウェアを可視化し、仮想ハードウェアとしてユーザに提供する。仮想 OS は、これに組み込まれているものとする。また、実際のユーザの立場は仮想化し、仮想ユーザとしてシステムに組み込む。これらによって、コンピュータ利用における自然な環境が実現できると考えた。そして、ユーザを OS になったつもりにさせることができる。

(2) 体験学習のできる環境にする

(1) の仕組みを実現するだけでは、体験と学習の仕組みが不十分である。

そこで本システムは、仮想ユーザが仮想 OS に要求を出すという行動を、ユーザにとっては問題が提供されたということにする。そして、その問題を解釈して対処するための仕組みを組み込み、ユーザが個人で学習できる環境を構築する。仮想ユーザには、ユーザの問題の答えへの正誤判断もさせる。これらによって本システムは、体験学習における学習の環境を実現できる。

また本システムは、扱う情報に関して、学習内容に関係ない部分はできるだけ省きたい。余分な情報を見せず、簡略に操作のできる学習の環境を実現するためである。またユーザに、体験、発見的学習のできる環境を提供したい。従来の学習にある、想像に頼る面を減らし、より深い理解の得られる環境を構築するためである。これらによって本システムは、体験学習における体験の環境を実現できる。

(3)OS の基本機能について学習させる

次の三点を体験学習させる内容の中心とする。

- プロセス管理
- メモリ管理
- ファイルシステム

この三点を組み込むことによって、OS の基本機能の学習を確保できると考えた。目標は、本システムを使えば、参考書などの補足資料を必要とせず、OS の中の限られた部分ではあるが、その学習内容を完結できるようにすることである。これによって、一つの学習環境として閉じたシステムを実現できる。また、従来の教科書や教師による OS の学習手段への補足や深い理解を提供できる環境を実現することも目標とする。

(4) 対象者は情報工学系の大学 2 年生以上とする

学習させる内容が情報処理の専門家が欲しい知識であること。また、その理解に予備知識が必要になることを考慮した。予備知識は、本システムで見せるものが、本来の OS の中では何を指しているものなのかを考えることのできる知識である。よって対象者は、国家試験の情報処理技術者試験程度の知識を持つユーザであり、情報工学系の大学 2 年生以上である。

(5) ユーザの操作内容を補足する

本システムは、学習内容と関係ない部分を省くと決めた。つまりユーザは、本来の OS に忠実でない振る舞いを体験することがある。そこで、そういった部分は補足する必要がある。

4 設計

システムの設計について述べる。

4.1 システムの全体構成

(1) 仮想ユーザと仮想ハードウェア

本システムは、設計方針の (1)、(2) を満足するために、仮想ユーザと仮想ハードウェアの二つで構成する (図 1 参照)。仮想ユーザは、ユーザに処理要求を提供する。処理要求とは、資源管理操作の体験学習における課題内容のことである。仮想ハードウェアは、ユーザに GUI コンポネントとグラフィックスを使って可視化したハードウェアを提供する。これは、ユーザが、そのデータ構造を操作するための GUI インタフェースも提供する。そして仮想ハードウェアのスペックは、ユーザの学習内容に必要な最低限のものを用意する。これらによって本システムは、体験学習モデルによる OS の学習環境を構築できると考えた。

(2) 仮想 OS

設計方針の (1) を実現するため、仮想 OS は、UNIX を想定して構成し、ユーザに UNIX のアルゴリズムに沿った資源管理操作の順序を期待する。そして本システムを使うユーザの行動は、仮想 OS の行う資源管理操作と等しいものとする。UNIX を想定する理由は、そのシステムが、長い歴史を経て完成されたシステムだからである。また、Windows のような過剰なサービスを持っていない。そのため、OS 自体が単純で筋の通った構造で実現されている。この特徴は、学習環境を実現したい本システムにおいて最適である。なぜなら学習において、分かりやすさは学習の効果をより一層高める。また、仮想 OS の具体的な操作内容は、ユーザ側で実装できるようにする。そして仮想 OS のデータ構造も、ユーザの学習内容に必要な最低限のものを用意する。これらによって本システムは、単純で筋の通った構造で、ユーザが頭を使う学習環境を構築できると考えた。

(3) 補足説明を表示する

設計方針の (5) を実現するため、ユーザの操作に対して補足説明を見せる機能を組み込む。ただし本システムは、設計方針の (2) で示したように、学習内容に関係ない情報を見せたくない。ここで文字列の説明を使うと、結局その実現を妨げる恐れがある。そのため、この補足の内容はできるかぎり短く構成する。

(4) ログを取る

ユーザが、ある課題に対して一通り学習していったとして、操作の途中の間違いを確認したい場合、その操作の履歴情報が必要になると考えた。そこで、タイムスタンプをつけたユーザの操作のログを取る。内容は、ユーザの選択した仮想ハードウェア上のデータ構造の名前や、設定した値などを取る。これによって本システムは、ユーザが自分の操作の履歴を確認できる仕組みを組み込むことができる。

(5) ユーザの基本的な操作の流れ

ユーザの基本的な操作の流れは、次の四つとする。

1. 処理要求プールから処理要求を選択する
2. 仮想ハードウェアと、その上に実現されたデータ構造を選択する
3. 処理要求の内容に沿った資源管理操作を行う
4. 資源管理操作の結果を仮想ユーザに渡す

ユーザは、これらの 1〜4 の操作を繰り返し行い、体験学習として OS の資源管理方式を学習する (図 1 参照)。つまり、1 で課題の選択、2 で解答手段の選択、3 で試行錯誤、4 で解答の四つの手順を実現する。

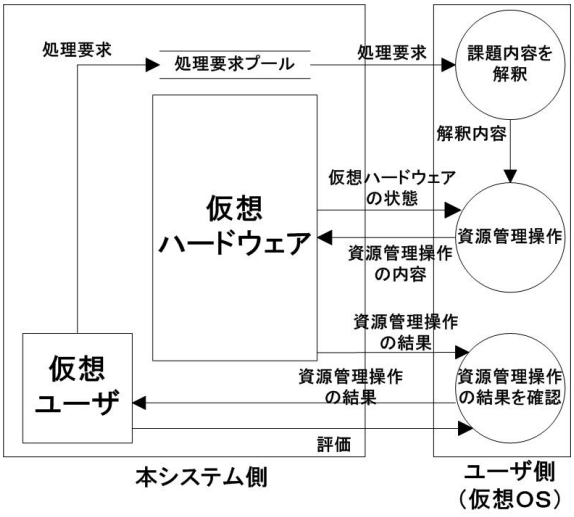

図 1: システムの全体構成

4.2 仮想ユーザの全体構成

仮想ユーザは、仮想の OS に、自由に、しかし意図を持って要求を出している。本システムは、この仕組みを GUI コンポネントを使って、問題の提供と正誤判断の機能を持たせることで実現する (図 2 参照)。目標は、実際のユーザが、実際の OS に要求を出すような仕組みを表現することである。

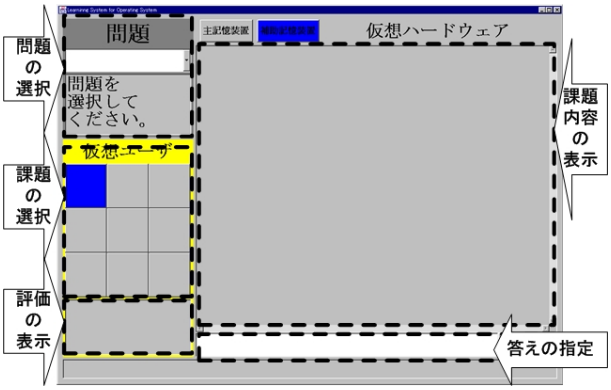

図 2: 仮想ユーザ

4.2.1 機能

仮想ユーザは、具体的に、次に示す機能を使って、問題の提供、正誤判断の仕組みを構成する。

(1) 課題の選択

ユーザが処理要求プールの中から、任意に処理要求を選択できるようにボタンを提供する。ボタンが押されたら、そのボタンと対応する課題の内容をテキストエリアに、また課題に対する答えを入力するためのテキストフィールドとボタンを表示する。これらによってユーザに、課題の選択と解答の手段を提供できる。

(2) 課題の提供

資源管理操作の課題は、課題一つに対してテキストファイル一つを用意し、その内容はシステムコールの集合で構成する (図 3 参照)。システムコールを採用した理由は、それ一つで、その目的を明確に示すことができるからである。これによってユーザに、自分の資源管理操作が具体的には何を解決しているのかを示す。

1 行目	課題の答え
2 行目以降	システムコール列

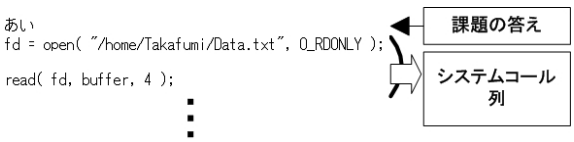

図 3: 課題ファイルのフォーマット

(3) 問題の提供

資源管理操作の問題は、問題一つに対してディレクトリ一つを用意し、その中には複数の課題のファイルを用意する。問題は例えば「バッファの役割」などの主題をディレクトリに名前を付けて用意して、学習内容の絞込みを行う。また、そのディレクトリ内には、課題を提供する順序を示すテキストファイルを一つ用意し、課題を提供する流れを作る (図 4 参照)。このファイルを順序ファイルと定義する。

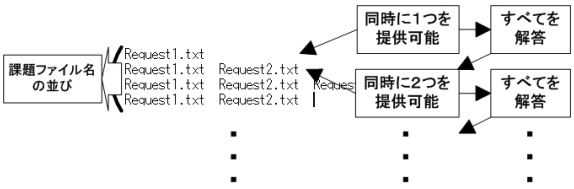

図 4: 順序ファイルのフォーマット

(4) 答えの指定

ユーザが、課題内容に沿った資源管理操作を終えて結果が分かったら、その結果が正しいかどうかを判定してもらうために仮想ユーザに伝える。答えを入力するためのテキストフィールドと、入力した答えに決定するボタンを用意し、答えの指定ができる手段を提供する。

(5) 評価

ユーザが資源管理操作の結果を渡してきたら、それが課題の答えと一致しているかどうかを判断し、ユーザの結果を評価する。評価は、ユーザの結果が課題の答えを満たしたか、満たしていないかによって「YES」、「NO」の二通りのメッセージを示す。本システムは、ユーザの操作の過程を評価しない。なぜなら設計方針 (2) で示したように、発見的学習のできる環境を実現したいからである。これらによって本システムは、正誤判断の仕組みを実現し、ユーザに、ユーザの結果の正解、不正解を伝える。

これらの五つの機能を実現し、図 5 のアルゴリズムを守ることにより、問題の提供、正誤判断の仕組みを実現する。

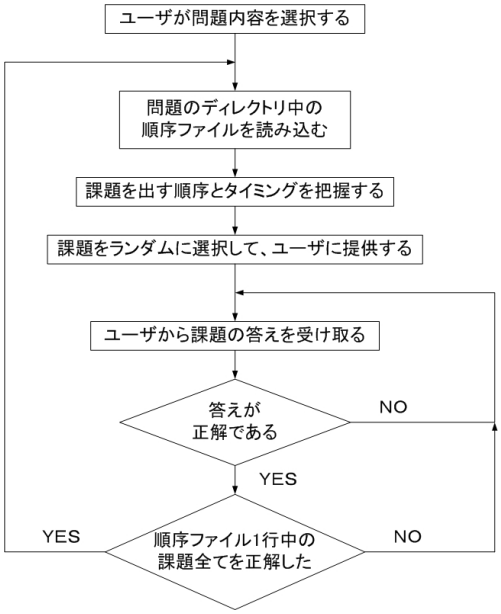

図 5: 仮想ユーザの問題提供、評価のアルゴリズム

4.3 仮想ハードウェアの全体構成

仮想ハードウェアは仮想 OS が組み込まれ、仮想 OS に資源管理を行われる。具体的には、GUI コンポネントやグラフィックスなどを使って、そのデータ構造を可視化した仮想主記憶装置と仮想補助記憶装置で構成される。

仮想主記憶装置とは、本来の主記憶装置を可視化したものであり、データ構造一つに対して一つのボタンを割り当て、それを押すと、そのデータ構造を操作するための GUI を表示する (図 6 参照)。具体的には、4.3.2 の (2)、(3)、(4) の仕組みを持ち、ユーザに資源管理操作をしてもらうために、そのインタフェースを提供している。

仮想補助記憶装置とは、本来の補助記憶装置を可視化したものであり、グラフィックを使った四つの同心円を用意して、それを選択すると、そのデータ構造を操作するための GUI を表示する (図 6 参照)。具体的には、4.3.2 の (1) の仕組みを持ち、ユーザに資源管理操作をしてもらうために、そのインタフェースを提供している。

4.3.1 機能

仮想主記憶装置のアドレス構造中の移動や、仮想補助記憶装置のブロック構造中の移動はボタンの選択により行う。また、仮想ハードウェアの状態の設定に GUI コンポネント (テキストフィールドやチェックボックス) を使う。これらによって、設計方針の (2) で示した、簡略に操作のできる環境の実現を行う。

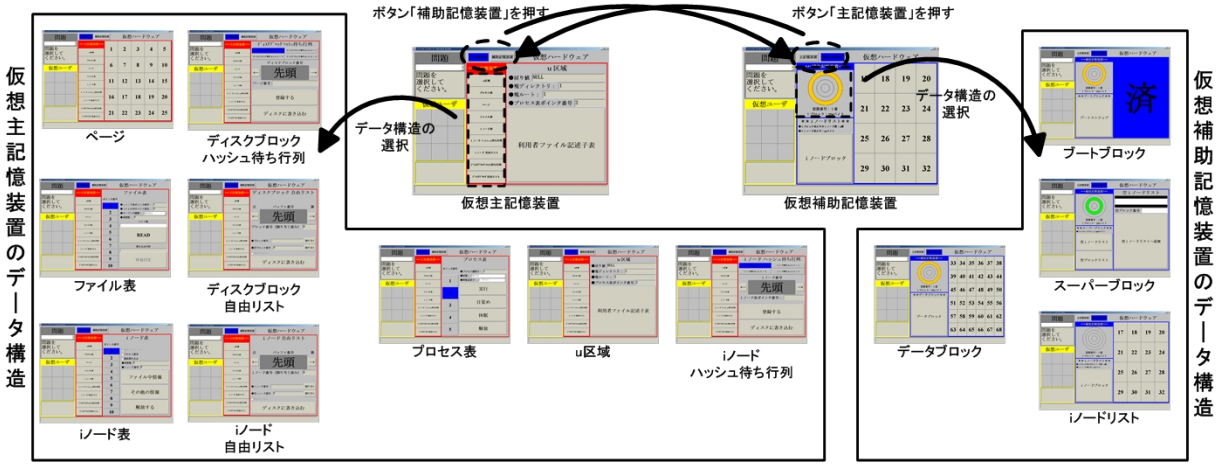

図 6: 仮想ハードウェア

4.3.2 可視化とシステムへの実装方法

本来の OS が持つデータ構造の中で何を選択し、どのように可視化するのかを次に述べる。

(1) ファイルシステム

ファイルシステム中のファイル構造の管理は、仮想 OS が UNIX を想定していること、また一貫性のとれた分かりやすい環境を実現することを考え、ｉノードの仕組みを採用する。

ファイルシステムを構成するデータ構造は、ブートブロック、スーパーブロック、ｉノードリスト、データブロックの四つを選択する。これによって、電源起動時の仕組み、ファイルシステムのキャッシュの仕組み、ｉノードとデータのブロックによる管理の仕組みを学習させることができる。

(2) バッファキャッシュ

バッファを管理するリストとバッファができることは、ボタンやチェックボックスなどを使って実現する。具体的には、ｉノードとディスクブロックにそれぞれ自由リストとハッシュ待ち行列を用意する。自由リストとは、未割り当てのバッファを集めて管理しているリストである。ハッシュ待ち行列とは、すでに情報が割り当てられたバッファを集めて管理しているリストである。本システムは、この二つのデータ構造をユーザに使わせて、バッファへのデータの割り当て、ディスク書き込み、解放の仕組みのタイミングと、その重要性を学習させる。GUI の状態の変化を利用することで視覚的な変化を与え、設計方針の (2) で示した環境の実現を行う。

(3) プロセス

本来のプロセスは、レジスタの値やテキスト、データなどの多くの情報を持っている。しかし今回、本システムは、操作の流れを仮想ユーザの提供する課題において管理することに決めた。またプロトタイプとして、資源管理操作の内容をファイルシステムとバッファとの間のデータのやり取りに限定した。このことから、プロセスは数値演算を行わない。また CPU の仕組みを実装しない。このことから、例外やコンテクストの仕組みを実装できない。よって、プロセスに時間の概念が必要ない。これらにより、本システムのプロセスは、必要な情報をプロセスの状態に限定することができる。

プロセスを構成するデータ構造は、想定した UNIX より、u 区域とプロセス表を選択した。これによって、プロセスの状態を管理する仕組みを実現できる。

(4) メモリ管理

想定した UNIX より、メモリの管理方式は、要求時ページングを選択した。データ構造は、ｉノード表、ファイル表を選択し、実装した。これらによって本システムは、ファイルに対して、open、close、

read、write を実行できるようにする。そして、ファイルへのデータの読み書き時のロックと参照数の関係の重要性とタイミングを学習させることができる。

5 実現

システムの実現について述べる。

5.1 実現

設計方針で挙げた二項目について、実現したことを次に述べる。

(1) ユーザを OS になったつもりにさせる

本システムは、この方針を実現するために、仮想 OS に UNIX を想定した。そして、そのデータ構造を可視化し、カーネルのできること全てはユーザのできることと等しいと決めた。今回はプロトタイプとして、本来の OS が持つ仕組みの中で実現した資源管理の内容は次の 4 点である。

1. ファイルシステム中の空ブロックの管理
   - スーパーブロックのキャッシュの仕組み
2. バッファの管理
   - 自由リスト、ハッシュ待ち行列を使ったキャッシュの仕組み
   - ディスクへの書き込み、遅延書き込みの仕組み
   - データの割り当て、解放、ロック、プロセス要求の関係の仕組み
3. プロセスの管理
   - プロセスの状態の切り換えの仕組み
4. システムコール
   - open、close、read、write の実行の仕組み

(2) 体験学習のできる環境にする

本システムは、この方針を実現するために、仮想ユーザの問題の提供、正誤判断の仕組みと、仮想ハードウェアのデータ構造を操作するための仕組みを実現した。

5.2 実現規模

本システムは Windows 上で、Java 言語を使って構築した。仮想ユーザは、設計であげた機能の全てを実現した。仮想ハードウェアは、データ構造ごとに管理を必要とすることを考慮し、データ構造に Java[2] のパッケージを設定し、必要なフィールドとメソッドを持たせて構成した。学習内容は、ファイルシステムとバッファとの間のデータのやり取りを中心に実現した。ソースコードの実現規模を表 1 に示す。

表 1: 構成ごとのソースコード行数

仮想ユーザ	３５０行
仮想主記憶装置	３５００行
仮想補助記憶装置	１５００行

5.3 操作例

ユーザが仮想 OS としてできることは図 7 に示す。

6 おわりに

本報告は、体験学習モデルによる OS の学習システムの実現をすることによって、ユーザに仮想化の仕組みを理解させる環境を提供することについて述べた。これによって本システムの仕組みは、OS の資源管理を学習することにおいて、効果的な手段であるということが分かった。今後の課題は、次の三点がある。

- より忠実な OS の仕組みを組み込んだ学習環境の実現
- 実際にユーザに使ってもらって、評価を取る
- ユーザの操作の過程を評価する仕組みを組み込む

参考文献

[1] Maurice J.Bach：UNIX カーネルの設計, 共立出版株式会社, 1991

[2] Joseph O'Neil：独習 Java, 翔泳社, 1999

[3] 結城 浩：Java 言語で学ぶデザインパターン入門, ソフトバンクパブリッシング株式会社, 2001

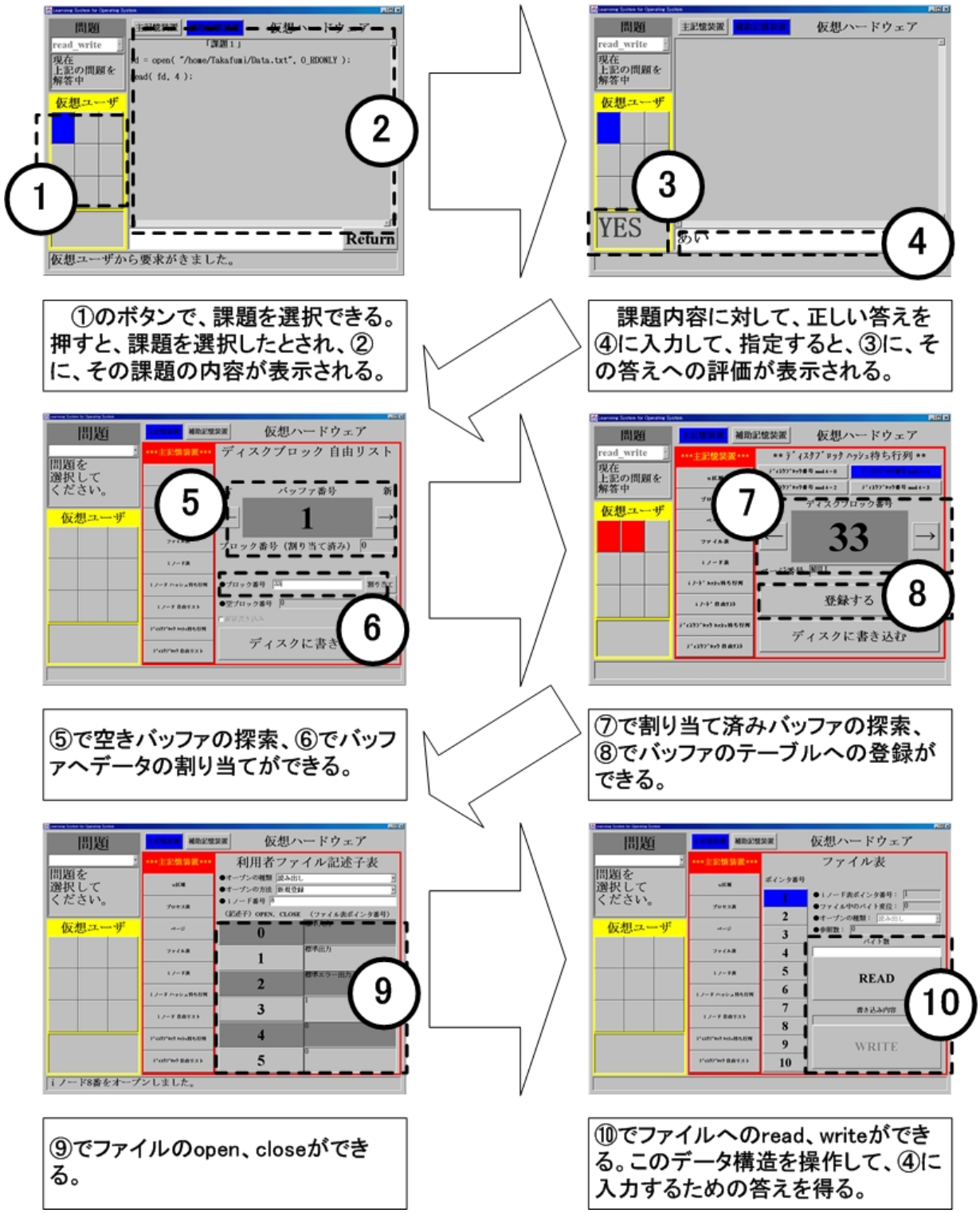
問題
read_write
現在
上記の問題を
解答中
仮想ユーザ
仮想ハードウェア
「課題1」
Return
仮想ユーザから要求がきました。
YES
①のボタンで、課題を選択できる。押すと、課題を選択したとされ、②に、その課題の内容が表示される。
課題内容に対して、正しい答えを④に入力して、指定すると、③に、その答えへの評価が表示される。
問題を
選択して
ください。
ディスクブロック 自由リスト
バッファ番号
ブロック番号（割り当て済み）
ディスクに書き
⑤で空きバッファの探索、⑥でバッファへデータの割り当てができる。
ディスクブロック番号
登録する
ディスクに書き込む
⑦で割り当て済みバッファの探索、⑧でバッファのテーブルへの登録ができる。
利用者ファイル記述子表
ファイル表
READ
WRITE
⑨でファイルのopen、closeができる。
⑩でファイルへのread、writeができる。このデータ構造を操作して、④に入力するための答えを得る。

図 7: 操作例