修 士 論 文

# 検索と類別表示を主とした情報管理手法の提案

北陸先端科学技術大学院大学
知識科学研究科知識システム基礎学専攻

梅津 亮

2001 年 9 月

# 目 次

# 第1章

# はじめに

個人的な情報管理を目的とした、HC というシステムを作ったので報告する。

本システムでは、個人的に得た情報やふと思いついたアイデア、メモを、カテゴライズせずにとにかく気軽にためておき、後刻それらを検索することで再利用に資する、という機能を提供することによって、個人の情報管理を向上させることを目的とするものである。

また同時に、PC の初心者にもインストールしやすいもの、ポータビリティの高いものにした。

# 第2章

# 研究の背景

前世紀末から、「情報化社会」ということが叫ばれて久しい。例えばアルビン・トフラーはその主著『第三の波』の中で、きたるべき情報化の波は、農耕の開始により余剰生産物が発生し、人口涵養力の増大、定住、交易、財産の私有がはじまるといういわゆる農業革命、そして 18 世紀イギリスの綿工業から起こって近代大衆社会を生みだした産業革命、これらと匹敵するほどの大きなものだと説いている。

しかし我々は、21 世紀を迎えた昨今、ますます増大する情報に囲まれながら、思いの外それらをうまく扱えていない、というのが現状である。例えば、ある事柄を知りたい、情報検索のシーンがある。確かにここ数年間の状況変化は目覚しく、WWW によって大統領の閨の睦言から MI6 の内情までいながらにして入手できるほどでになった。また、PageRank に基づいた検索結果のランキングを提供する Google の登場により、ユーザーの検索要求は大分満たされるようになってきている。

だが、その検索方法の主流は依然としてキーワード投入方式であり、当然のことであるが、検索要求に見合った適切なキーワードをまず思いつく必要がある。ある程度の長さを持つ比較的特殊な用語の What タイプの検索要求であれば、キーワードの選択並びに該当情報の発見は割合容易であろう。が、「いいかもしんまい」という流行語のルーツであるとか、ソフトウェアのインストール時に遭遇したトラブルの回避法など、When や How、Why といったタイプの検索要求となると、検索キーワードの選択にはたと困ることが多いのではなかろうか。もちろんこういった点に対して、自然言語ベースの検索エンジンも登場しつつあるが、まだその精度は多くの検索要求を満たすほどには至っていない。

他方、このように折角 WWW から得た情報をどこに保存しておくか、というのも問題になってくるだろう。従来あまりこういったことには光が当てられてこなかったきらいがあるが、上述のような情報化社会に対処していくには避けられない点であると思われる。例えば、とあるソフトの設定方法がわからなかった場合を想定する。WWW を猟渉し、結果、見つけたとする。その URL 及び書かれている内容をどうするだろうか。もちろん、最初はそれをオンラインでブラウズしながら利用するだろう。だが、後でまた同じ、もしくは類似の問題に遭遇するとい

うのは非常によくあることである。殆んどの場合、その内容や URL はもとより、そこへの到達の仕方、またどんなキーワードをどんな順番で用いて検索したかなどは覚えているものではない。

そこで当然、普通は必要な情報を見つけた時点でそれを何らかの形で保存しておくことになろう。問題になるのは、どのようなメディアにどうやって情報を保存しておけば、再利用性が高く利便性が良いか、ということである。

よく付箋紙に書いてモニターに貼りつける人を見かける。しかしそれでは、ほんの当座に見直す場合ならともかく、時間の経過とともに情報が散逸しやすく、紙媒体であるので検索も効率的ではない。当該ページを印刷して保存しておくケースもよくあるであろうが、これも結局紙媒体であるので、付箋紙の場合と同じような問題を抱えることになる。

一方、ブックマークに追加するというのは電子的な保存に繋がり、上述のような難点は回避されるやに思えるものの、そもそもが「効率的な情報の保存・再利用」に向けられたものではないため、自分でカテゴライズしなければならず、またブックマークを検索することもできないものである。カテゴライズというのは、分類に困るものが出現しやすく、ラベリングを人手で施すのは骨が折れるものである。できたとしても、膨大なブックマークの海から深い階層をたどって行くことになるであろう。

このように、情報検索後の情報の保存管理法一つとってみても、人によって工夫はあるかもしれないが、多くの場合、効率的な情報管理はできていないといってよい。

こういったことは、メールを介して受け取った有用な情報を管理する場合についても同じことがいえる。後で参照したい内容があったとしても、MUA の検索機能を駆使したところで、膨大なメールから探し出すのに困難が伴うことが多い。また、探す対象がブックマークやメールや付箋紙といったように分散してしまうと、検索効率はますます落ちることになる。

## 2.1 他の情報マネージング

WWW やメールなどから情報を得る場合だけでなく、例えば本を読んで書き留めたいフレーズなどに出会ったときや、ふとしたアイデアを思いついたときなど、これもそういった情報をどこでどのように管理するかという問題が生じる。

そのような発想法ともいうべきようなものの類は、これまでにあまた世に現出している。ただしそのほとんどは、「カードに書き留めよ」「溜ったカードを整理せよ」といったように、紙メディアに対するオペレーションが中心で、情報過多の現代に対応するものであるとはいえない。ただしカードのような紙媒体でも、KJ 法のようにすぐその場で使用するなら、検索・散逸はさほど問題にならないだろう。

## 2.2　電子メディアによる情報整理手法

『「超」整理法』(野口 1993) に示されるような、無整理時系列順の個人情報管理手法を説くものがある。そもそも、情報を分類することはできないといいきり、その理由として、以下を挙げている。

- こうもり問題 (どの項目に入れるべきかが不明瞭)
- その他問題 (分類できないものが膨れ上がる)
- 誤入問題 (誤った項目に分類する)
- 分店時の在庫引き継ぎ問題 (項目を細分化する時の書類の処置)
- < 君の名は > シンドローム (分類した項目名を忘れる)

従って、本稿でもこの「敢えて分類しない」という路線を継承することにする。

## 2.3　人間の特性との相補性

人間の記憶は非常に気が利いていて、必要なことはすぐに思い出せるし、様々な意味で類似の概念を想起することができる。しかし同時に忘れやすい。一方、コンピューターの提供する記憶は、人間のそれと全く逆である。即ち、原則的には丸暗記であるが、決して忘れることはない。

人間は、複数の概念・対象の類似性の判定は極めて柔軟にできる。しかし、情報をブラウジングするのは、概して苦手である。一方コンピューターは、類似性の判定に柔軟性はないが、一度電子化してしまえば、そのデータをブラウジングするのは容易である (櫻井 1995)。

以上の相補的な特徴を利用して、人間の記憶能力・ブラウジング能力を実質的に高め、人間の情報創造活動・情報利用活動に資しようというのも目的の一である。

## 2.4　知識の重要性

そもそも、どうして個人ベースの情報管理が必要か、という点であるが、世の趨勢もさりながら、再びアルビン・トフラーの言を引くならば、彼の近著『パワーシフト』では、権力の源泉である「暴力・富・知識」を『第三の波』における文明区分に当てはめ、「第三の波」の到来によりパワーのバランスが知識に集中し始めたという。このような指摘を待つまでもなく、その重要性はいや増しに高まっているといえよう。従って、このような個人ベースの情報管理ツールの登場も期待されるものと考える。

# 第3章

# 関連ツール

こういった情報管理の非効率性を解消するものとして、既に様々なツールが世に出ている。それらの幾つかを概観していく。

## 3.1 memoma

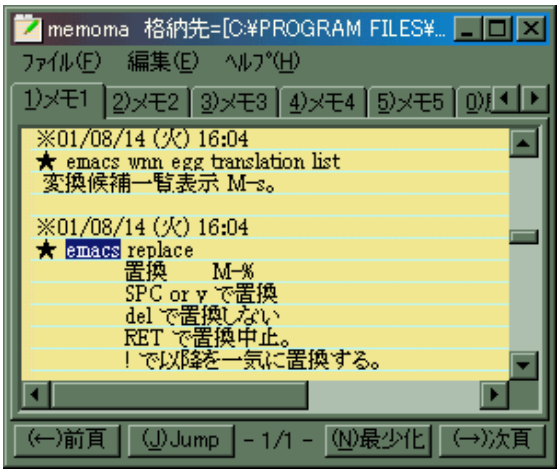


図 3.1: memoma

Windows 上のメモソフト (フリーソフト) である。起動すると、図 3.1 のようなウィンドウがポップアップし、日付と時刻の下にメモの内容を入力するシステムになっている。「メモ」は数種類用意され、また簡単な (1 語のキーワードによる逐次) 検索機能もついている。

しかし、メモが数種類に分かれており、これを利用しようとする場合には、入力時に分類を要求するのと同じことであり、どれをどう分類したらよいのか、という問題がたちまちのうちに生じよう。気付いた、もしくは調べたメモ・情報の入力時の敷居は低くするべきであり、このように、入力時に分類を要求する形の

システムでは、ふとしたアイデアも何番のメモに書くか考えるなどの余計な思考を必要とし、入力障壁が高いと言わざるを得ないだろう。

## 3.2 付箋紙 2000

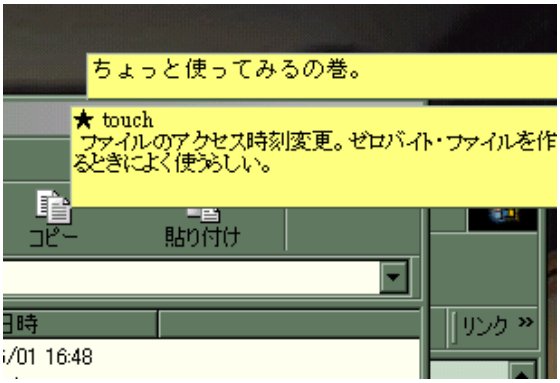


図 3.2: 付箋紙 2000

図 3.2 に示すのは、windows 上のアプリケーションである「付箋紙 2000」である。

これは、図 3.2 に見られるように、物理的な付箋紙と同じように、黄色い紙片に覚えておきたいメモを書き込み、それをデスクトップに貼付するものである。暫時書き留めておきたいデータに対しては有効であろうが、長期的な保存になると、この点においても物理的な紙媒体と同じように、他のアプリケーション・ウィンドウに散逸、または埋もれてしまい、わかりにくくなるとともに、検索機能もなく、有効な再利用性あるとはいいがたい。折角電子媒体なのであるから、紙媒体の模倣ではなく、電子媒体の利点を活かす方向性を出すべきではなかろうか。

## 3.3 Q-Pocket

図 3.3 の Q-Pocket は、ソニー・コンピュータサイエンス研究所の増井氏による情報管理手法である。「あらゆる情報を ID だけで管理すること及び既存の情報の検索/更新/情報追加を基本操作とする」(増井 2000) というコンセプトであり、本稿のものに近い。データ形式も単なるプレインテキストなので、他方面での利用という応用も利く。

しかし、その実装から窺われる思想を考えると、その中心となるのは図 3.3 のようなメモのリストアップ画面であり、折角エディターである Emacs 上にクライアントを実装したのに、編集画面からの接続が悪くなっている。検索も、KAKASI を用いた「動的曖昧検索」、即ち漢字をローマ字の読みがなでインクリメンタルに

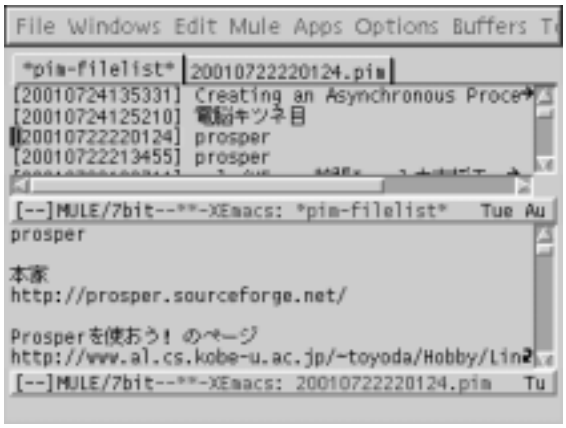


図 3.3: Q-Pocket

検索するもので、同音異義語の多い日本語文章の検索には向いていない。検索して使うことが主眼になる筈であるが、実装上からその考慮が手薄であるように窺われる。

また、サーバー-クライアント式としてクライアント側の実装に自由を持たせたとのことだが、Q-Pocket のサーバーはローカルマシンで使用するのにポートを開ける形となっていたり、クライアントの方はバッファーが例えば沢山開かれるなど、使い勝手の良いものとはいえない。インストール方法もコンパイルを必要とする他、満足なドキュメントが附属していないため、導入するに至るまでには多大な労力を必要としてしまっているのが残念である。

## 3.4 MHC


図 3.4: MHC

MHC(図 3.4) はスケジュール管理ソフトであるので、個人情報蓄積・検索ソフ

トとはやや趣を異にするが、Emacs ベースで MUA とともに動くという注目すべき点があるのでここに引いた。

スケジュールは主にメールで通知されることに着目して、Emacs ベースの MUA(Mew、Wanderlust、mh-e、GNUS) で読んでいるメールから、簡便な操作で予定に落とすことができることを売りにしている。従って、そのデータ形式は MUA と同じ MH 形式 (1 メール 1 ファイル) である。

また Unix において、多くのユーザーが最も接している時間が長いアプリケーションは Emacs であろうし、更に Emacs 上の MUA を開いている時間が長いであろう、という点から、MUA との統合 (寄生) のアイデアは優れているといえよう。

勿論、スケジューラーという性格上、アイデア、メモをため込んだり検索したりするには (MUA の検索機能を使うとしても) 不便である。

# 第4章

# 設計

本ツールは、仮に HC と名付ける。

以上に述べた従来の電子情報管理手法の欠点を補うために、HC では以下のような特徴を持つものとする。

## 4.1 未分類の入力

ふとした情報・メモを入力していくためには、入力に対する敷居が低くなくてはならない。入力する度に、そのデータをどのカテゴリーに入れるかを考えたり、適当な、かつユニークなファイル名やどのディレクトリー階層に置くか、といったことを全く意識させないようにする必要がある。

本ツールでは、ファイル名は自動的に連番を振るようにし、ディレクトリー構造は敢えてフラットとする。各メモファイルは ID だけで管理し、基本的には『「超」整理法』(野口 1993) に説かれているように、敢えて雑多な情報を未分類のまま時系列順に保持・整理・提示する。

## 4.2 データ形式

メモデータファイルは、形式に制約を設けず、様々なソフトから利用可能なように、プレインテキストを旨とする。また、コンピューターに向かっている場合、エディターを使っている時間が長いと思われるので、クライアントとして Emacs インターフェイスをまず用意した。エディターのうちでも、MUA を使用していることが多いであろうということ、並びにメモのリスト閲覧画面が MUA のメール閲覧画面と似ていることから、将来の MUA との統合を鑑み、データ形式は MH 形式を採用した。

MH 形式とは即ち、1 メモ 1 ファイルであって、日本語は JIS コードで書かれており、メモファイルのタイムスタンプは「Date:」フィールドを、タイトルには「Subject:」フィールドを用いることをいう (「From:」は無意味だがつけても害は

ない)。これでメールをそのままメモファイルとして扱っても整合性が取れる。

## 4.3 サーバー-クライアント方式

Q-Pocket と同じようにサーバーとクライアントを分けて、様々なクライアント形態から使用できるようにする。今回は Emacs Lisp によるクライアントの実装しか用意しなかったが、これだと Meadow などを用いない限り Windows ユーザーにとっては導入が難しいので、Java ベースのクライアントも作る予定である。

また、サーバーの方はポートをオープンするものではなく、コマンドラインベースとした。ANSI C の範囲内の C 言語で記述したので、Solaris、Linux だけでなく Windows でも動くなど、ポータビリティも維持されている。

## 4.4 検索結果のクラスタリングによる分類表示

メモの検索結果が数十に及んだ場合、それらがクラスターとして内容ごとにまとまった形で表示されれば、より検索しやすいと思われる。そこで、検索結果が一定数以上の場合、クラスタリングを施すことを考えた。

新規メモが追加された場合、その都度単語・文書行列を作成し、その行列に対して特異値分解を施し、一定次元数に縮退した行列を用意しておく (LSA:潜在的意味解析については後述)。そうして、検索キーワードが投入されたときに、その検索キーワードを含む文書の単語ベクトルだけを縮退した行列から抽出して、それらのベクトルの類似度をそのユークリッド距離によるものとし、nearest-neighbour 法によってクラスタリングすることとした。

## 4.5 潜在的意味解析

クラスタリングする際の類似度計算のためのベクトル生成には、潜在的意味解析 (LSA:Latent Semantic Analysis)(Deerwester,Dumais,Landauer and Harshman 1995) を用いた。

潜在的意味解析とは、単語の各文書における共起情報を元に、特異値分解を行なって次元を圧縮することで、文書間の相互関係を浮かび上がらせる手法である。

以下に簡単な例を示してその効果を説明する。

例えば、以下のような単語・文書行列があるとする。

これは、縦の d1〜d6 が文書、横が単語で、数字は出現頻度を表している。「cosmonaut」は文書 d1 と d3 に 1 回ずつ出現する。

この文書・単語行列を、特異値分解を用いて、2 つの正規直交行列と特異値が降順に並んだ 1 つの対角行列に分解する。一般に、行数 $M$ が列数 $N$ 以上である任意

| | d1 | d2 | d3 | d4 | d5 | d6 |
|---|---|---|---|---|---|---|
| cosmonaut | 1 | 0 | 1 | 0 | 0 | 0 |
| astronaut | 0 | 1 | 0 | 0 | 0 | 0 |
| moon | 1 | 1 | 0 | 0 | 0 | 0 |
| car | 1 | 0 | 0 | 1 | 1 | 0 |
| truck | 0 | 0 | 0 | 1 | 0 | 1 |

の $M \times N$ 行列 $A$ は、

$$A = U\sigma V^T$$

という形で書けることが知られている (北 1999)。

上述の単語・文書行列を特異値分解すると、以下のような行列になる。左上から $U$、$\sigma$、下段が $V$ である。

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.44 | 0.23 | -0.57 | -0.58 | -0.25 | 2.16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.13 | 0.33 | 0.59 | -0.00 | -0.73 | 0.00 | 1.59 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.48 | 0.51 | 0.37 | -0.00 | 0.61 | 0.00 | 0.00 | 1.28 | 0.00 | 0.00 | 0.00 |
| 0.70 | -0.35 | -0.15 | 0.58 | -0.16 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| 0.26 | -0.65 | 0.41 | -0.58 | 0.09 | 0.00 | 0.00 | 0.00 | 0.00 | 0.39 | 0.00 |
| | | | | | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.75 | 0.29 | -0.28 | 0.00 | 0.53 | 0.00 | | | | | |
| 0.28 | 0.53 | 0.75 | -0.00 | -0.29 | 0.00 | | | | | |
| 0.20 | 0.19 | -0.45 | -0.58 | -0.63 | -0.00 | | | | | |
| 0.45 | -0.63 | 0.20 | 0.00 | -0.19 | 0.58 | | | | | |
| 0.33 | -0.22 | -0.12 | 0.58 | -0.41 | -0.58 | | | | | |
| 0.12 | -0.41 | 0.33 | -0.58 | 0.22 | -0.58 | | | | | |

ここで、2 次元に縮退する。これは、元の行列の最小 2 乗法による最適近似になっている。なお、文書間の比較をする場合、$U$ は不要なので、以下 $\sigma$ と $V$ について記す。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2.16 | 0 | 0.75 | 0.28 | 0.20 | 0.45 | 0.33 | 0.12 |
| 0 | 1.59 | 0.29 | 0.53 | 0.19 | -0.63 | -0.22 | -0.41 |

この行列を掛け合わせ、ユークリッド距離に基づく類似度を取ると、以下のような対称行列になる。

ここで、元々の文書・単語行列を思い浮かべると、文書 d2 と文書 d3 には共通して出現する単語がなかったにもかかわらず、ここでは文書 d2 と d3 の類似度は

| | d1 | d2 | d3 | d4 | d5 | d6 |
|---|---|---|---|---|---|---|
| d1 | 1.00 | 0.78 | 0.95 | 0.47 | 0.74 | 0.11 |
| d2 | 0.78 | 1.00 | 0.94 | -0.18 | 0.16 | -0.53 |
| d3 | 0.95 | 0.94 | 1.00 | 0.18 | 0.49 | -0.20 |
| d4 | 0.47 | -0.18 | 0.18 | 1.00 | 0.94 | 0.93 |
| d5 | 0.74 | 0.16 | 0.49 | 0.94 | 1.00 | 0.75 |
| d6 | 0.11 | -0.53 | -0.20 | 0.93 | 0.75 | 1.00 |

0.94 と、高い値を示している。これは、d2 に出現する単語と d3 に出現する単語がそれぞれ文書 d1 に出現し、文書 d1 が媒介したためである。

また同様に、文書 d5 と d6 にも共通して出現する単語がないにもかかわらず、その類似度は 0.75 と高い値を示している。これも、文書 d4 の方に、d5 に出て来る「car」、d6 に出てくる「truck」の双方が出てきているからである。

## 4.6 クラスタリングの効果

この部分を今回の提案システムに組み込む前に、試験的にその効果を試してみた。客観的、定量的にその有効性を表わすことは難しいのだが、例えば「make」でメモを検索した場合、44 個の検索結果が得られた。「wmaker」など部分文字列にもマッチしているため、「wmaker」関係はそれでひとまとまりになっていたら検索しやすくなろうというものであるが、人の目による各クラスターの結束性を確認できなかった。他のキーワードについてもほぼ同様で、有効な結果を示せないようであった。また、特異値分解には時間がかかるので (約 800×8600 の行列に対し、Sun Enterprise 3500[UltraSPARCIIs400MHz, メモリー 1.5GB] で約 1 時間)、今回の提案システムには実装することを見合わせた。

# 第5章

# 使用の実際

## 5.1 基本的な使用法

現在のところ Emacs クライアントのみの提供であるので、それに沿って説明する。

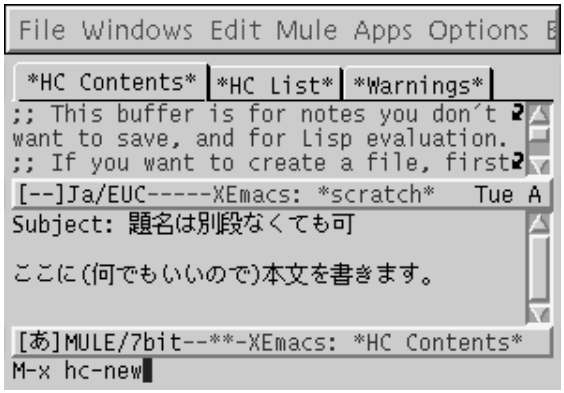


図 5.1: 新規メモの追加

ユーザーは基本的には、Emacs に向かってメールを読み書きしたり、プログラムをコーディングしていたり、といった編集作業中である場合が多いだろう。そこで、例えばメールで友人から「これこれのようなサイトがある」という URL を受け取ったとしよう。どうも有用そうである。本当に有用であるかどうか、将来再度検索するかどうか、もしかするともう一生見ないかもしれないデータにすぎないかもしれない。けれども、将来の有用性を今現在で判断するのは主旨ではない。とにかくどんどんデータを投入するべきであり、そういった結果として「無駄」なデータがあったとしても、昨今のディスク資源、メディアのバイト当たり単価を考えれば、全く意に介さないものである。それよりはとにかくデータを蓄積するようにする。あとで何が必要になるかはわからないものである。

そこで、図 5.1 のように「M-x hc-new」と打つと、メモ入力用のウィンドウが

出現し、そこに必要な情報を書き込む。あとは、普通のファイル保存と同じように「C-x C-s」と打てば、ウィンドウは消滅し、内容はメモデータとして自動的にナンバリングされ、「Date:」情報がなければ付与され、メモデータとして格納されることになる。これだけである。


図 5.2: search 画面

次に、「あの情報はどこへやったっけなぁ」という検索の場面である。図 5.2 のように、いついかなる編集場面からでも、「M-x hc-search」と打ち込み、そして検索語句を投入すれば、その語句 (and 検索) を含むメモがリストアップされてくる。あとはそれを順次閲覧して、希望の情報を得ればよい。

## 5.2 応用例

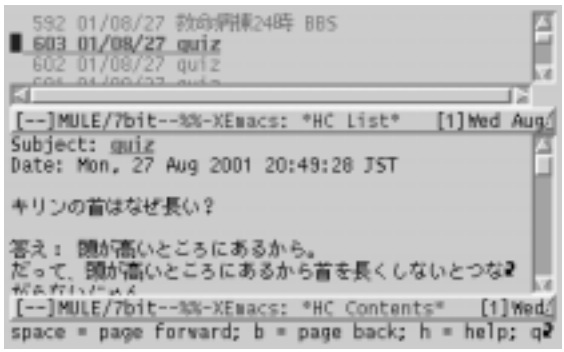


図 5.3: メモのリスト画面

リスト画面から更に続けて検索をしたい場合には、「s」と打てばよいようになっている。「S」と大文字で打ち込んだ場合は、「Subject」フィールドだけに対する選択的な検索をかけることができる。

リストアップの順番は、時系列順になっている。これは『「超」整理法』(野口 1993) 指摘の通りであるが、検索がきちんと行なわれて完全に絞り込まれていれば、時系列順に並んでいる必要性すらなくなっている。

メモの内容に不備があったり、変更があったりした場合には、「e」と打てば当該メモを編集することができる。編集前のデータを破棄しなければ、具体的には編集メモ保存時に「C-u C-x C-s」で保存した場合には、異なる ID を持つ複数のデータが保存されることになるので、バージョン管理の必要がなくなる。古い情報を消さずに保存したい場合に用いる。

また、「D」で当該メモの消去を行なう。誤って消去しないよう大文字のコマンドにして、誤操作を防いでいる。「o」でメモファイルの再ナンバリングを自動的に行なっているので、連番の欠番は生じないようになっている。「g」で最新リストの再描画を行なうので、直近のメモを閲覧するのに便利である。

最後に、「q」で HC のリストを終了することができる。HC を起動する前のウィンドウ状態に復帰するので、HC を気軽に開始・終了することができる。

こういったキー・バインディングは「h」を押すことによって出現するヘルプでも確認することができる。

## 5.3 導入

本ツールで重点を置いたことの一つとして、ポータビリティの高さ並びにインストールの簡便さを保つ、ということがある。実際、導入に際しては、サーバーの方は gcc で `hc.c` をコンパイルするのみ、クライアントの方は `.emacs` にメモファイルを置くディレクトリーと、`hc.c` の実行ファイルの場所を示すだけでよいようになっている。一般の Windows ユーザーにとっては Emacs の使用自体が障壁になるが、これは別にクライアントを作って対処する予定である。

# 第6章

# 結論

個人的な情報管理を向上させることを目的とした個人情報管理システムを提案し、HC という形で設計、実装を行なった。カテゴライズしない簡便な入力、後刻の検索による蓄積された情報の再利用、を具現したインターフェイスを実装し、同時に高いポータビリティ低いインストール障壁も持ち合わせることができた。

クライアントの今後の方向性としてまず考えられるのが、検索効率の向上である。例えば、表記の揺れを超えた検索を行なえるようにする、より具体的には、LSA で単語の類似度を計算し、類似度が高くかつ相互排他的に出現する (同一文書には共起しない) ものを拾えば、特定できるのではないかと考えている。こうしたことにより効率的な検索を支援できれば、知識の再利用、再生産により貢献するとともに、ネットワーク化などで検索対象が拡大した時にも、目的の情報にアクセスしやすいであろう。

また、従来のユーザーのメール使用頻度並びにメーラー (MUA) の完成度を鑑み、MUA との統合、寄生をはかることも考えられる。Emacs ベースの多くの優れたメーラーは、既に洗練されたファイル表示機能や、(簡易ではあるが) 検索機能を持っている。それらの機能を利用しない手はないであろう。

サーバーの方としては、ネットワーク化が挙げられる。こういったメモデータをネットワークに出してもいいものなら、Gnutella のように、クライアント同士を繋いでメモデータを共有することも可能となる。そうなってくると、「知の共有」という形態へ向かうことができる。

他方、既存の WWW にも検索の範囲を広げるべきかもしれない。その際、データ形式は XML にするのが好ましいだろう。ハイパーリンクを使えれば、メモデータ内での関連性情報をも持たせることができる。

このように世界を広げていくことで、個人ベースの知識ノウハウが共有できる場を作ることができよう。

# 謝辞

常に鋭い指摘で指導を行なわれ、研究室を主宰して下さった櫻井教授、心配を頂きながらアドバイスをして下さった荒木助手、サブテーマを指導して頂いた石崎助教授、授業を通じて多くの啓発を頂いた小長谷教授と林助教授に感謝する。

　また、個人ベースの情報管理に関して、多大なアイデアとアドバイスを頂いた原田哲治氏には特に名を記して謝したい。

# 参考文献


櫻井彰人 (1995). “研究テーマの検討 (15).” 日立中央研究所.

Brin, S. and Page, L. (1998). “The Anatomy of a Large-Scale Hypertextual Web Search Engine.” *Computer Networks*, **30**.

Deerwester, S.,Dumais, S.T. an Furnas, G.,Landauer, T.,and Harshman, R. (1995). “Indexing by latent semantic analysis.” *Journal of the American Society for Information Science*.

北研二 (1999). 確率的言語モデル. 言語と計算. 東京大学出版会.

増井俊之 (2000). “検索と例情報を活用した情報管理手法 Q-Pocket.” インタラクティブシステムとソフトウェア VIII.

野口悠紀雄 (1993). 「超」整理法. 中公新書. 中央公論社.

# 付 録A

# program list

```
/* Personal Memo handler server command. */

/* Copyleft (C) 2001 by u-ryo <u-ryo@jaist.ac.jp> */

/* $Id: hc.c,v 1.5 2001/09/19 16:22:29 u-ryo Exp $ */

/* This file is free software; you can redistribute it and/or modify */
/* it under the terms of the GNU General Public License as published by */
/* the Free Software Foundation; either version 2, or (at your option) */
/* any later version. */

/* This file is distributed in the hope that it will be useful, */
/* but WITHOUT ANY WARRANTY; without even the implied warranty of */
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*/
/* GNU General Public License for more details. */

/* You should have received a copy of the GNU General Public License */
/* along with GNU Emacs; see the file COPYING.  If not, write to */
/* the Free Software Foundation, Inc., 59 Temple Place - Suite 330, */
/* Boston, MA 02111-1307, USA. */


#define _GNU_SOURCE

#include <stdio.h>
#include <dirent.h>
#include <string.h>
```

```
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/stat.h> /* for statbuf */

/* #define CYGWIN */

int files_size = 500;

struct memo_properties {
  char name[1024];
  char title[1024];
  time_t date;
};

struct for_sort {
  char new_file_name[1024];
  char old_file_name[1024];
  int flag, visit_flag;
};

typedef struct memo_properties MEMO_PROPERTIES;
typedef struct for_sort FOR_SORT;

int f(MEMO_PROPERTIES *v1, MEMO_PROPERTIES *v2) {
  if (v1→date < v2→date)
    return 1;
  else if (v1→date > v2→date)
    return −1;
  return (0);
}

int iseuc(unsigned char c) {
  if (c ≥ 0xa1 && c ≤ 0xfe)
    return 1;
  return 0;
}
```

```
/* http://kanji.zinbun.kyoto-u.ac.jp/~yasuoka/kanjibukuro/japan.html */
void jis2euc(char *buff, char *buff2) {
  int i = 0, mode = 0, ii = 0;

  while (i < strlen(buff)) {
    if (buff[i] == 27) {
      if (buff[i+1] == 36 && buff[i+2] == 66) {
        mode = 1;
        i += 3;
      } else if (buff[i+1] == 40 &&
                 (buff[i+2] == 66 || buff[i+2] == 74)) {
        mode = 0;
        i += 3;
      }
      continue;
    }
    buff2[ii++] = buff[i++] + (mode ? 128 : 0);
  }
  buff2[ii] = '\0';
}

void euc2jis(char *buff, char *buff2) {
  int i = 0, mode = 0, ii = 0;

  while (i < strlen(buff)) {
    if (iseuc(buff[i])) {
      if (mode == 0) {
        buff2[ii++] = 27;
        buff2[ii++] = 36;
        buff2[ii++] = 66;
        mode = 1;
      }
    } else {
      if (mode == 1) {
        buff2[ii++] = 27;
        buff2[ii++] = 40;
        buff2[ii++] = 66;
        mode = 0;
      }
```

```
    }
    buff2[ii++] = buff[i++] - (mode ? 128 : 0);
  }
  buff2[ii] = '\0';
}

char *pathfinder(char *full_file_name) {
  char *cptr, *file_name;
  cptr = full_file_name;
  cptr += strlen(full_file_name);
  while (cptr > 0) {
    cptr--;
    if (*cptr == '/') {
      *cptr = '\0';
      file_name = cptr + 1;
      return file_name;
    }
  }
  return NULL;
}

void check_dir(DIR **dir, char *dir_name) {
#ifdef CYGWIN
  if (strrchr(dir_name, '/') ≠ NULL &&
      strlen(strrchr(dir_name, '/')) == 1)
    dir_name[strlen(dir_name)-1] = '\0';
#endif
  if ((*dir = opendir(dir_name)) == NULL) {
    printf("Can't open %s.\n", dir_name);
    perror("Can't open dir");
    exit(-1);
  }
  if (strrchr(dir_name, '/') == NULL ||
      strlen(strrchr(dir_name, '/')) ≠ 1)
    strcat(dir_name, "/");
}

int read_file(DIR *dir, FILE **fp, char *dir_name, char *fname,
              time_t *date) {
```

```
  struct stat statbuf;
  struct dirent *dp;
  int i = 0;

  if ((dp = readdir(dir)) == NULL)
    return 0;
  stat(dp->d_name, &statbuf);
  if (S_ISDIR(statbuf.st_mode))
    return read_file(dir, fp, dir_name, fname, date);
  while (i < strlen(dp->d_name))
    if (!isdigit(dp->d_name[i++]))
      return read_file(dir, fp, dir_name, fname, date);
  strcpy(fname, dir_name);
  strcat(fname, dp->d_name);
  *date = statbuf.st_mtime;
  if (NULL == (*fp = fopen(fname, "r"))) {
    printf("Can't open file %s\n", fname);
    exit(-1);
  }
  return 1;
}

void substring(char *str, int from, int to) {
  int i;

  if (from ≥ strlen(str) || to > strlen(str) || from > to)
    return;
  for (i=from; i<to; i++)
    str[i-from] = str[i];
  str[i-from] = '\0';
}

void check_title(char *buff, char *title, int *title_flag) {
  if (strncmp(buff, "Subject: ", strlen("Subject: ")) == 0) {
    strcpy(title, buff);
    substring(title, strlen("Subject: "), strlen(title));
    *title_flag = 1;
  }
}
```

```
void check_date(char *buff, time_t *date, int *date_flag) {
  struct tm *time_ptr, timestruct;

  if (strncmp(buff, "Date: ", strlen("Date: ")) == 0) {
    time_ptr = &timestruct;
    strptime(buff, "Date: %a, %d %b %Y %T", time_ptr);
    *date_flag = 1;
    *date = mktime(time_ptr);
  }
}

void check_from(char *buff, int *from_flag) {
  if (strncmp(buff, "From: ", strlen("From: ")) == 0) {
    *from_flag = 1;
  }
}

int digitp(char *buff) {
  int i = 0;

  while (i < strlen(buff))
    if (!isdigit(buff[i++]))
      return 0;
  return 1;
}

/* http://www.google.com/search?q=cache:QgCKpUWlA_w:oasis2.openave.net/pub/88/6/lib.strstr.c.
*/
/* char *strstr(Source, What) */
/*      register const char *Source; */
/*      register const char *What; */
/* { */
/*   register char WhatChar; */
/*   register char SourceChar; */
/*   register long Length; */

/*   if ((WhatChar = *What++) != 0) { */
/*     Length = strlen(What); */
```

```
/*     do { */
/*       do { */
/*         if ((SourceChar = *Source++) == 0) { */
/*           return (0); */
/*         } */
/*         if (iseuc(*Source)) */
/*           *Source++; */
/*       } while (SourceChar != WhatChar); */
/*     } while (strncmp(Source, What, Length) != 0); */
/*     Source--; */
/*   } */
/*   return ((char *)Source); */
/* } */

int my_strstr(char *source, char *what) {
  int i=0, j=0,
    src_len = strlen(source),
    what_len = strlen(what);

  do {
    do {
      if ((*(source+i+j) == *(what+i)) ||
          (((*(what+i) ≥ 97 && *(what+i) ≤ 122 &&
            *(source+i+j) == *(what+i)−32) ||
           (*(what+i) ≥ 65 && *(what+i) ≤ 90 &&
           *(source+i+j) == *(what+i)+32))))
        i++;
      else
        break;
      if (i == what_len)
        return 1;
    } while (i < what_len);
    if (iseuc(*(source+i+j)))
      j++;
    j++;
    i = 0;
  } while (j ≤ src_len − what_len);
  return 0;
}
```

```
void search(char *buff, int argc, char *argv[], int *found) {
  int i = 0;

  for (i=3; i<argc; i++)
    if (my_strstr(buff, argv[i]))
      found[i-3] = 1;
}

void sort(struct for_sort *new_and_old, int i) {
  int counter = 0;
  char fname[512];

  new_and_old[i].visit_flag = 1;
  if (new_and_old[i].flag == 1)
    return;
  if (fopen(new_and_old[i].new_file_name, "r") ≠ NULL)
    while (1) {
      if (!strcmp(new_and_old[i].new_file_name,
                  new_and_old[counter].old_file_name)) {
        if (new_and_old[counter].visit_flag == 1) {
          tmpnam(fname);
          rename(new_and_old[i].new_file_name, fname);
          //printf("%s => %s %i\n",new_and_old[counter].old_file_name, fname, i);
          strcpy(new_and_old[counter].old_file_name, fname);
          break;
        }
        sort(new_and_old, counter);
        break;
      }
      counter++;
    }
  new_and_old[i].flag = 1;
  rename(new_and_old[i].old_file_name, new_and_old[i].new_file_name);
  //printf("%s => %s %i\n",new_and_old[i].old_file_name, new_and_old[i].new_file_name,i);
}

void ls(int argc, char *argv[]) {
  DIR *dir;
```

```
FILE *fp;
char buff[8192], buff2[8192], fname[512], dir_name[1024], title[1024],
  *filename, buff3[8192];
struct memo_properties *memo_prop;
int (*cmp)() = f;
int i = 0, counter = 0, date_flag = 0, title_flag = 0, found[argc-3];
struct tm *time_ptr;
time_t temp_date;
struct for_sort *new_and_old;

strcpy(dir_name, argv[2]);
check_dir(&dir, dir_name);
memo_prop = (MEMO_PROPERTIES *)malloc(files_size *
                                      sizeof(MEMO_PROPERTIES));
//buff3 = (char *)malloc(sizeof(char));
while (read_file(dir, &fp, dir_name, fname, &temp_date)) {
  title_flag = 0;
  date_flag = 0;
  title[0] = '\0';
  buff3[0] = '\0';
  while (fgets(buff, 8192, fp) ≠ NULL) {
    if ((!strcmp(argv[1], "ls") || !strcmp(argv[1], "sort")
         || !strcmp(argv[1], "title_search"))
        && title_flag && date_flag)
      break;
    //buff2 = malloc(sizeof(char) * strlen(buff));
    jis2euc(buff, buff2);
    if (!title_flag)
      check_title(buff2, title, &title_flag);
    if (!date_flag)
      check_date(buff, &temp_date, &date_flag);
    if (!strcmp(argv[1], "search")) {
      if (buff[0] == '\n') {
        search(buff3, argc, argv, &found[0]);
        //free(buff3);
        //buff3 = (char *)malloc(sizeof(char));
        buff3[0] = '\0';
      } else {
        //buff3 = (char *)realloc(buff3, sizeof(char) *
```

```
        //(strlen(buff2) + strlen(buff3)));
        strcat(buff3, buff2);
        //if (iseuc(buff2[strlen(buff2)-2]))
        if (buff3[strlen(buff3)−1] == '\n')
          substring(buff3, 0, strlen(buff3)−1);
        if (!iseuc(buff3[strlen(buff3)−2]))
          strcat(buff3, " ");
      }
    }
    //free(buff2);
  }
  search(buff3, argc, argv, &found[0]);
  fclose(fp);
  if (!strcmp(argv[1], "title_search"))
      search(title, argc, argv, &found[0]);
  for (i=3; i<argc; i++)
    if (found[i−3] == 1)
        found[0]++;
  if (((!strcmp(argv[1], "search") || !strcmp(argv[1], "title_search")) &&
       found[0] == (argc − 2)) || !strcmp(argv[1], "ls") ||
      !strcmp(argv[1], "sort")) {
    if (counter == (counter % files_size))
      if (!(memo_prop =
            (MEMO_PROPERTIES *)realloc(memo_prop,
                                       (counter + files_size) *
                                       sizeof(MEMO_PROPERTIES))))
        printf("%s\n", "Short of memory.");
    strcpy(memo_prop[counter].name, fname);
    memo_prop[counter].date = temp_date;
    strcpy(memo_prop[counter].title, title);
    counter++;
  }
  for (i=3; i<argc; i++)
    found[i−3] = 0;
}
closedir(dir);

if (counter > 0)
  qsort(memo_prop, counter, sizeof(MEMO_PROPERTIES), cmp);
```

```
  if (!strcmp(argv[1], "sort")) {
    date_flag = 0;
    new_and_old = malloc(counter * sizeof(FOR_SORT));
    for (i=1; i≤counter; i++) {
      strcpy(dir_name, memo_prop[counter−i].name);
      filename = pathfinder(dir_name);
      if (i ≠ atoi(filename)) {
/*        strcpy(new_and_old[date_flag].old_file_name, dir_name); */
/*        strcat(new_and_old[date_flag].old_file_name, "/"); */
/*        strcat(new_and_old[date_flag].old_file_name, filename); */
        strcpy(new_and_old[date_flag].old_file_name, memo_prop[counter−i].name);
        sprintf(filename, "%d", i);
        strcpy(new_and_old[date_flag].new_file_name, dir_name);
        strcat(new_and_old[date_flag].new_file_name, "/");
        strcat(new_and_old[date_flag++].new_file_name, filename);
      }
    }
    for (i=0; i<date_flag; i++)
      sort(new_and_old, i);
  }

  if (!strcmp(argv[1], "ls") && argc > 3 && digitp(argv[3]))
    if (counter > atoi(argv[3]))
      counter = atoi(argv[3]);
  if (strcmp(argv[1], "sort"))
    for (i=0; i<counter; i++) {
      time_ptr = localtime(&memo_prop[i].date);
      printf("%s %02d/%02d/%02d %s", memo_prop[i].name,
            time_ptr→tm_year−100, time_ptr→tm_mon+1,
            time_ptr→tm_mday, memo_prop[i].title);
    }
  if (counter == 0)
    printf("Not_found.\n");
}

void cat(char *file_name) {
  FILE *fp;
  char buff[8192], buff2[8192];
```

```
  //buff2 = malloc(sizeof(char) * strlen(buff));
  if (NULL == (fp = fopen(file_name, "r"))) {
    printf("Can't open %s\n", file_name);
    exit(1);
  }
  while (fgets(buff, 8192, fp) ≠ NULL) {
    jis2euc(buff, buff2);
    printf("%s", buff2);
  }
  //free(buff2);
}

void rm(int argc, char *argv[]) {

  unlink(argv[2]);
  //pathfinder(argv[2]);
  //ls(argc, argv);
}

void max_name(char *dir_name, char *fname) {
  DIR *dir;
  FILE *fp;
  time_t temp_date;
  char *temp_name;
  int counter = 0;

  check_dir(&dir, dir_name);
  while (read_file(dir, &fp, dir_name, fname, &temp_date)) {
    temp_name = pathfinder(fname);
    if (counter < atoi(temp_name))
      counter = atoi(temp_name);
    fclose(fp);
  }
  closedir(dir);
  sprintf(fname, "%d", (counter + 1));
}

int is_same_file(FILE *fp1, FILE *fp2) {
```

```
  size_t size1, size2;
  char buff1[8192], buff2[8192];

  fseek(fp1, 0L, SEEK_END);// move a file pointer to last
  size1 = ftell(fp1);// get a file size
  fseek(fp1, 0L, SEEK_SET);// move a file pointer to first
  fseek(fp2, 0L, SEEK_END);
  size2 = ftell(fp2);
  fseek(fp2, 0L, SEEK_SET);
  // return false(0) if different size
  if (size1 ≠ size2)
    return 0;
  // check contents
  while (fgets(buff1, 8192, fp1) ≠ NULL) {
    fgets(buff2, 8192, fp2);
    if (strcmp(buff1, buff2))
      return 0;
  }
  return 1;
}

int is_there_same_file(char *dir_name, FILE *fp1) {
  DIR *dir;
  FILE *fp2;
  time_t temp_date;
  char fname[512];

  check_dir(&dir, dir_name);
  while (read_file(dir, &fp2, dir_name, fname, &temp_date)) {
    if (is_same_file(fp1, fp2)) {
      fclose(fp2);
      closedir(dir);
      return 1;
    }
    fclose(fp2);
  }
  return 0;
}
```

```
void copy(FILE *from, char *to) {
  FILE *fp, *temp_fp;
  char buff[8192], buff2[8192], title[1024], fname[512], temp_name[100];
  int title_flag = 0, date_flag = 0, line_num = 0,
    from_flag = 0;
  time_t temp_date;
  struct tm *tm_ptr;

  tmpnam(fname);
  /* mkstemp(char *template) returns file discripter.
     I can't get a temporary file pointer.
     And mkstemp man tells "Don't use this function, use tmpfile(3) instead."
     But tmpfile(3) returns temporary file pointer only for WRITE. */
  temp_fp = NULL;
  if (NULL == (temp_fp = fopen(fname, "w"))) {
    printf("Can't open file %s\n", to);
    exit(1);
  }
  while (fgets(buff, 8192, from) ≠ NULL) {
    euc2jis(buff, buff2);
    if (!title_flag)
      check_title(buff2, title, &title_flag);
    if (!date_flag)
      check_date(buff2, &temp_date, &date_flag);
    if (!from_flag)
      check_from(buff2, &from_flag);
    fputs(buff2, temp_fp);
  }
  fclose(temp_fp);
  if (NULL == (temp_fp = fopen(fname, "r"))) {
    printf("Can't open file %s\n", fname);
    exit(1);
  }
  if (NULL == (fp = fopen(to, "w"))) {
    printf("Can't open file %s\n", to);
    exit(1);
  }
  line_num = 0;
  while (fgets(buff, 8192, temp_fp) ≠ NULL) {
```

```
    if (title_flag == 0 && line_num == 0)
      fputs("Subject: ", fp);
    fputs(buff, fp);
    if (date_flag == 0 && line_num == 0) {
      time(&temp_date);
      tm_ptr = localtime(&temp_date);
      strftime(temp_name, 100, "Date: %a, %d %b %Y %H:%M:%S JST\n", tm_ptr);
      /* if (temp_name != NULL) */
        fputs(temp_name, fp);
    }
    if (from_flag == 0 && line_num++ == 0) {
      fputs("From: ", fp);
#ifdef CYGWIN
      fputs("winuser@localhost\n", fp);
#else
      fputs(getenv("LOGNAME"), fp);
      gethostname(temp_name, 128);
      strcat(temp_name, "\n");
      fputs(temp_name, fp);
#endif
    }
  }
  fclose(fp);
  fclose(temp_fp);
  unlink(fname);
}

void save(char *dir_name, FILE *file) {
  char fname[512];

  max_name(dir_name, fname);
  strcat(dir_name, fname);
  copy(stdin, dir_name);
}

void merge(char *dir1_name, char *dir2_name) {
  DIR *dir;
  FILE *fp;
  char fname[512], destination_file[512],
```

```
    from_file[512], destination_dir_name[512];
  int counter = 0;
  time_t temp_date;

  max_name(dir2_name, destination_file);
  counter = atoi(destination_file) - 1;
  strcpy(destination_dir_name, dir2_name);
  check_dir(&dir, dir1_name);
  while (read_file(dir, &fp, dir1_name, from_file, &temp_date)) {
    if (!is_there_same_file(destination_dir_name, fp)) {
      strcpy(destination_file, destination_dir_name);
      sprintf(fname, "%d", ++counter);
      strcat(destination_file, fname);
      copy(fp, destination_file);
    }
    fclose(fp);
  }
  closedir(dir);
}

/* あとがき */

/* 2001年8月30日14時半、1時間に及ぶ修論発表会が終わりました。 */
/* 本当は30分の予定だったんですけど、この時期は発表者が立て込んでないので、
*/
/* みっちりやってもらってしまいました。 */
/* 「研究」とは程遠い、「こんなの作ってみました」程度のものでしたので、 */
/* 発表中は、穴があったら入りたく、なければ掘ってしまいたかったです。 */

/* 「接してきた時間が少ないから間違ってるかもしれないけど、 */
/* きみは大きな研究をやろうとして、結局ダメになってるような気がするね。 */
/* もっと小さなことでも凄いんだなぁっていうことは沢山あるから、 */
/* そういうのをもっと大事にしないと。 */
/* ソニーの増田さんの研究だって、簡単に×つけるんじゃなくって、 */
/* あれインストール簡単にすれば全部○になっちゃうでしょう? */
/* それでいいじゃないと思っちゃうんだけど」 */

/* 心理分析までしてもらえました。 */
/* 自分でも確かに、そんな気はします... */
```

/* 追い詰められてしまう程、 */
/* 大逆転を狙わねばならないという $pressure$ に襲われてしまうんです。 */
/* でも、そんな大それたことは当然簡単にはできません。 */
/* 結局切羽詰まって、お茶を濁してしまうという $pattern$、という感じです。 */

/* 「中間発表の時から考えると$...$」 */

/* 林先生が、「そら、ご覧」という顔を一瞬、されます。 */
/* こちらも、 */
/* 前日の発表練習で「中間発表と違う理由を聞かれるかもしれないから */
/* 一応答えられるようにしておいた方がいいよ」と林先生に言われたのに、 */
/* 答弁を用意してなかったので焦りましたが、続く言葉に救われました。 */

/* 「よく半年でここまでまとめたね。努力は認めるよ」 */

/* 少し悔しいですけどね。 */
/* これをやりたくて $JAIST$ 来たわけじゃないですし、単なる努力賞ですし。 */
/* ただ、あぁ言われた後だけに$...$ */

/* 「これだけのものに $LSI$ を使うなんて大袈裟だよ。 */
/* 使うとしても、単純に頻度でやるんじゃなくて、 */
/* 頻度を正規化$?$してからやるとか、しないと。 */
/* サブテーマでやらせた $LSI$ がまさかここで使われるとは思わなかった」 */

/* 流石は専門家。 */
/* 敢えて $stop\ list$ を用いずに、頻度でやってみたんですけどね。 */
/* $1\ memo$ 当たりの単語量が少ないのがうまくいかない原因かなぁと */
/* 思っておりましたが、 */
/* やりようによっては何とかなるんでしょうかね。 */
/* 何とかなるんなら何とかしたかったですが、 */
/* いかんせん時間がありませんでした。 */

/* 次は、小長谷先生からです。 */

/* 「一言でこのソフトのウリを説明すると、何になるかな$?$」 */

/* 詰まってしまいました。こんなところで。 */
/* 一応、『入力時無分類、検索再利用の簡便性、 */

/* の組み合わせ』という感じなんですけど、 */
/* あまりにも弱いのは否めません。 */

/* 「個人情報システムはどうあるべきか、 */
/* とかもっと大上段に構えてもよかったんじゃないかな。 */
/* 同じモノでも、見せ方によって変えるとかね」 */

/* なるほど、そっか。 */
/* けどしかし、そもそもこれやる予定じゃなかっただけに、 */
/* そこまで大きく *survey* もしてませんでした。 */

/* 「何かこれの有効性を客観的に示した実験はやらなかったの*?*」 */

/* 「それは、時間がなかったので*...*」 */

/* ホント、 */
/* 初めからこれをやるつもりだというわけではなかったので、 */
/* 「時間がなかった」の *on parade* になってしまいました。 */
/* 小長谷先生は次にお世話になる先ですので、 */
/* もっと格好のつくものをと心配して下さったことだと思います。 */
/* この日も朝、一旦自宅に帰る折にたまたま校門付近で出くわし、 */
/* 開口一番、 */

/* 「ソースをつけなさい」 */

/* 「は」 */

/* 「短過ぎる」 */

/* 「すいません*!*」 */

/* 単純に量の問題でもないような気はするんですが、 */
/* 質に関してもなってないだけに、何もいえません。 */
/* 櫻井先生と *interaction* する暇がまるでなく、 */
/* とにかく「日本語で埋めた」だけですから*...* */
/* *(*大雑把な *story* はゼミで発表して承認受けまたと思われますけどね*)*。 */

/* 林先生からは、 */

/* 「server と client に分けたのは、 */
/* user にとって必ずしも使いやすいものとは限らないわけで、 */
/* data 形式だけ揃えてそれぞれ別個の application を作った方が */
/* 良かったんじゃないかな」 */

/* そうですね... */
/* server-client(といっても port 開くわけではないので */
/* 正確には server-client 形式とはいえないのですが、 */
/* このように分けたのは、確かに単なる開発者側の都合です。 */
/* 多様な client を抱えても大丈夫なように、ということなんですけれど、 */
/* data 形式を揃えれば十分ではないか、というのは、 */
/* 実は当たっていると思います。 */
/* しかも現在は Emacs client しか用意していないので、 */
/* 分けたのは事実上『無意味』になってしまっています。 */
/* 林先生からは、 */

/* 「実際の開発の現場では、user の要求を聞かないと...」 */

/* 云々という follow もあり、一応ことなき?を得ました。 */

/* 櫻井先生からは、 */

/* 「server と client を分けるというのは...」 */

/* と、林先生が自分への反論だと思ったらしく、 */

/* 「いぇ、私のいいたいのは...」 */

/* どちらの言いたいこともわかりますので、こちらとしてはまぁまぁ、 */
/* という感じだったんですが、 */

/* 「それはそれとして、mailer との統合というのは新しい面だと思うんですよね。
*/
/* もっと色々苦労した点を盛り込めば良かったんじゃないかな」 */

/* 先生すいません、折角の御指摘ですが、mailer 統合も原田さんの idea です。 */
/* というより MHC でしょうか。 */

/* audience は、午後開催ということもあってか、 */

```
/* K1・2後ろ半分が埋まる程度でしょうか、来てしまっていました。 */
/* 始まる直前研究室には誰もいず、 */
/* 櫻井先生心配してらした程でしたが(私は嬉しかったのですけど)、 */
/* 直行した人が多かったようです。 */
/* 恥ずかしくってaudienceはあんまりよく見てなかったんですが、 */
/* 中程に座ってた原田さんが、 */

/* 「そうそう、よくぞ言ってくれた!」 */

/* というような満面笑顔で大きくうなずいていたのは見えました。 */
/* あと、河崎おねぇさまが途中であまりの下らなさに出ていかれたくらいでしょうか。 */

/* 終了後、いつまでもK1・2から立ち去れなかったのは、 */
/* shockのせいもありますが、 */
/* projectorをoffに出来なかったのが主因です。 */


int main(int argc, char *argv[]) {
  if (argc < 2) {
    printf("Usage: %s ls dir_name [max_number_for_list_up]\n", argv[0]);
    printf("       %s [title_search|search] dir_name keywords...\n", argv[0]);
    printf("       %s [cat|save] filename\n", argv[0]);
    printf("       %s sort dir_name\n", argv[0]);
    printf("       %s rm filename [max_number_for_list_up]\n", argv[0]);
    printf("       %s merge from_dir to_dir\n", argv[0]);
    exit(-1);
  } else if ((strcmp(argv[1], "search") || (strcmp(argv[1], "title_search"))) &&
       argc > 3 && argv[3][strlen(argv[3])-1] == '"')
    substring(argv[3], 0, strlen(argv[3])-1);
  if (!strcmp(argv[1], "search") ||
      !strcmp(argv[1], "ls") ||
      !strcmp(argv[1], "sort") ||
      !strcmp(argv[1], "title_search"))
    ls(argc, argv);
  else if (argc > 2 && !strcmp(argv[1], "cat"))
    cat(argv[2]);
  else if (argc > 2 && !strcmp(argv[1], "rm"))
    rm(argc, argv);
```

```
  else if (argc > 2 && !strcmp(argv[1], "save"))
    save(argv[2], stdin);
  else if (argc > 3 && !strcmp(argv[1], "merge"))
    merge(argv[2], argv[3]);
  return 0;
}
```