平成 17 年度　卒業論文

コンピュータ教室混雑状況表示プログラムの提案

文教大学　情報学部　経営情報学科

氏名　柴田　紘明

目次

概要

湘南校舎には、800 台と 14 のコンピュータ教室があるが、レポート提出期間やメディアセンターの１階にあるコンピュータ教室のようなユーザが集中しやすいコンピュータ教室では、混雑がみられる。現在、コンピュータ教室の混雑緩和への取り組みは行われていないが、空き教室の情報を提供することで、ユーザを分散させ、また、ユーザが空いている教室を探す手間を省くことができるのではないかと考えた。この研究では、コンピュータ教室へ集中するユーザを分散させる仕組みについて考えていく。

コンピュータ教室混雑状況表示プログラムの提案

文教大学　情報学部　経営情報学科
柴田紘明　a2p21085

１．はじめに

私は、大学の情報処理課でアルバイトをしていた。その業務の中で、教室を巡回する作業があるが、その際、教室の混雑にはばらつきがあるように感じた。現在、湘南校舎では、ユーザの混雑の緩和への取り組みは、とくに行われていないが、ユーザが利用したいソフトウェアや、ハードウェアのある教室の情報を提供することで、ユーザが、わざわざ教室へ足を運ばなくても、自分が利用したい PC 教室の利用状況が分かるようなモノがあれば良いのではないかと考えた。

湘南校舎の情報システムでは、ユーザのログオンとログオフのデータをログとして保存している。普段このログは、特に問題が起こらない限り分析されることはない。しかし、ログからログオンとログオフの情報を引き出すことで、コンピュータ教室の利用状況の表示に利用できるのではないかと考えた。

ログには、ログオンとログオフの情報だけが出力されているわけではなく、ログオンとログオフの判別には、必要でない情報も出力されている。そのうえ、ログオンとログオフの情報は、ログに明確にログオンとログオフが記録されているわけではなく、ログを読込み、そこから文字列を切り出してログオンとログオフを判別させなければならなかった。また、入力データとして利用した１つ１つのログは膨大で、それら全てをメモリ上に展開し分析するのには、やや無理があった。

そこで、この研究では、手間のかからない分析の方法を考え、コンピュータ教室の利用状況を湘南校舎のコンピュータシステムを利用する、すべてのユーザに提供したい。また、実際にログを利用した、プログラムを作成する。

2 章では、湘南校舎のコンピュータ教室の混雑の現状、過去の取り組みや既存のソフトウェアについて説明し、3 章では作成したプログラムの説明をする。4 章では、結果と考察。そして、今後の課題について述べる。

２．背景

ここでは、作成したプログラムで解決する問題とその背景について述べる。

2.1 現状

湘南校舎には、学生が利用できるコンピュータ教室が 20 教室あり、その教室には、849 台の PC がある。また、コンピュータ教室には、授業のない時間は自由に利用できる授業用教室が 15 教室と、いつでも自由に利用できるオープン利用教室が 5 教室ある。普段はあまり、混雑はみられない。しかし、レポートや課題の提出期間などは、よく混雑がみられるが、現在は、ユーザを分散させるための取り組みは行われていない。

2.2 過去の問題への取り組み

湘南校舎の情報システムが現在のものに入れ替わる以前、7101 教室の前にあるディスプレーで利用状況の表示が行われていた。以下に以前、利用状況の表示のために、利用されていたウェブページを載せた。

Shonan INformation CEntRE

PC教室利用状況

Room	Idle	Use	Down	State	利用率	利用率グラフ	Room
7101	5	17	0	自由利用	77%		7101
7201	21	3	16	自由利用	13%		7201
7401	16	0	0	自由利用	0%		7401
7402	0	0	16	Down	0%		7402

Oct7_11:24_Update

図 1．以前、提供されていた利用状況表示システムの表示画面

これは、ログオンとログオフの情報をデータベースにためておき、そこから 20 分おきにデータベースにためこんだ情報を取得し、ウェブサイトとして表示させている。
誰もログオンしておらず、PC が待機中であれば“Idle”、だれかが PC にログオンしていれば “Use”、PC の電源が切れていれば “Down”のところに、それぞれ数字が表示される。教室自体の電源が切れていれば”State”の項目にやはり”Down”と表示される。そして、利用率はパーセンテージと横の棒グラフで表示される。

シンプルで分かりやすい機能を持ったシステムだったが、このシステムは、コンピュータ教室を利用するユーザのログオンの情報の取得が、現在の湘南校舎の情報システムで導入しているユーザの情報を管理するシステムに対応していない。そのため、上図のシステムでは、ログオンの情報をデータベースにためることができず、現在は機能していない。

2.3 プログラムの提案

利用状況表示システムは、現在は機能していないが、まだ存在している。しかし、現在のユーザ情報を管理するシステムに対応しておらず、ログオンの情報を取得することができないので、現在は利用することができない。

2003 年のシステム入替えが行われる以前は、ユーザ情報の管理にActiveDirectory*1を使用していた。ActiveDirectoryでは容易にログオンの情報を取得することができた。しかし、入替えが行われてからは、Samba*2というシステムをユーザの情報の管理に利用するようになり、ログオンの情報を容易に取得することができなくなった。

そこで、AcitveDirectory で容易に実現できていたことを Samba でも実現し、利用状況表示システムに取り込めるような形式に出力するプログラムを提案したい。

３．入力データ

ここでは、入力データとして使用する Samba のログについて説明する。そして、このログの持つ問題点について説明していきたい。

3.1　ログの構造

Samba のログには、コンピュータの利用状況やデータ通信の記録が出力されている。また、その記録。操作やデータの送受信が行われた日時と、行われた操作の内容が記録される。そのため、ログか必要な部分を読み取り、ユーザのログオン・ログオフの判別をしなければならない。まず、以下の例を見ていただきたい。

```
[2004/11/30 09:15:17, 1] smbd/service.c:make_connection(629)
  s3107010 (172.20.20.10) connect to service netlogon as user a2p21085 (uid=18911, gid=105) (pid 17852)
```

ユーザがコンピュータにログオンあるいはログオフすると、このような 2 行のデータがログに記録される。そして、ログの 1 行目には時間、2 行目にはコンピュータ番号、IP アドレス、ユーザ ID、pid が記録されている。pid はユーザがコンピュータにログオン・ログオフしたときにだけ出現する。また、pid は通し番号になっており、翌日には新しく pid が割り当てられる。そのため、pid でログオンとログオフの判別も行う。

次に、ログオンとログオフの判別の仕方について述べる。まずログオンの場合から説明していく。

以下は、ユーザがログオンした際のデータである。

```
[2004/11/30 09:15:17, 1] smbd/service.c:make_connection(629)
  s3107010 (172.20.20.10) connect to service netlogon as user a2p21085 (uid=18911, gid=105) (pid 17852)
[2004/11/30 09:15:17, 1] smbd/service.c:make_connection(629)
  s3107010 (172.20.20.10) connect to service netlogon as user a2p21085 (uid=18911, gid=105) (pid 17852)
```

ユーザがコンピュータにログオンすると、それぞれ、同じ「日時」「コンピュータ番号」「IP アドレス」「ユーザ ID」「pid」が書かれた 2 行が、連続して 2 回記録される。次に、ログアウトする場合について説明する。下図は、ユーザがログアウトしたときのデータである。

```
[2004/11/30 09:15:58, 1] smbd/service.c:make_connection(629)
  s3107010 (172.20.20.10) connect to service netlogon as user a2p21085 (uid=18911, gid=105) (pid 17873)
```

ユーザがコンピュータからログアウトする場合は、「日時」「コンピュータ番号」「IP アドレス」「ユーザ ID」「pid」が書かれた 2 行が 1 回記録される。このとき、pid はログオン時のものとは違う pid が新しく割り当てられる。そのため、ログオンとログオフの判別は pid で行うことができる。

3.2　ログの問題点

以下の例を見て欲しい。

```
[2005/10/31 13:23:38, 1] smbd/service.c:make_connection_snum(662)
  s3107012 (172.20.20.12) connect to service netlogon initially as user a2p21085 (uid=18911, gid=105) (pid 23554)
[2005/10/31 13:24:09, 1] smbd/service.c:make_connection_snum(662)
  s3107012 (172.20.20.12) connect to service netlogon initially as user a2p21085 (uid=18911, gid=105) (pid 23554)
[2005/10/31 13:25:09, 1] smbd/service.c:make_connection_snum(662)
  s3107012 (172.20.20.12) connect to service netlogon initially as user a2p21085 (uid=18911, gid=105) (pid 23554)
[2005/10/31 13:25:22, 1] smbd/service.c:make_connection_snum(662)
  s3107012 (172.20.20.12) connect to service netlogon initially as user a2p21085 (uid=18911, gid=105) (pid 23554)
[2005/10/31 13:27:24, 1] smbd/service.c:make_connection_snum(662)
  s3107012 (172.20.20.12) connect to service netlogon initially as user a2p21085 (uid=18911, gid=105) (pid 23554)
```

これは、ログの一部をまとめたものである。同じ pid を持つ 2 行が出現するのは、通常は、ログオンのときに 2 回だけだが、まれに、2 回以降も同じ pid を持った 2 行が出現する場合があった。その場合は、pid だけでは判別することができない。そのため、このようなパターンが出現したときのために、例外処理を行う必要がある。だが、今回は例外処理の作成を行うことはできなかった。

また、ログは、1 つのファイルとしてまとまっているのではなく、湘南校舎にある 5 台のサーバにそれぞれ分散されており、ログオンの情報や、ログオフの情報もそれぞれ 5 台のサーバにバラバラに保存されている。そのため、1 つのサーバに保存されているログにログオン・ログオフの情報が記録されているとは限らない。

4．プログラムへの実装

3 章では、入力データとして利用するログについて説明をした。ここでは、ログを用いて

実際にどのようなプログラムを作成したかを説明する。

4.1 プログラムの説明

作成には、以前、授業で使用したことのある C 言語を使用した。ここでは、プログラムに実装できたログオンとログオフの判別機能について説明していく。また、プログラムの詳細については、付録に載せたソースコードを参照されたい。[1] [2]

このプログラムは、ログからログオンとログオフを判別する。プログラムを起動すると、そのため、ログを 2 行づつ読み込み、pid がある行を探す。pid を見つけたら、その行から「時間」「マシン番号」「ユーザーID」「pid」を切り出し、動的に記憶領域を確保したノードに格納する。

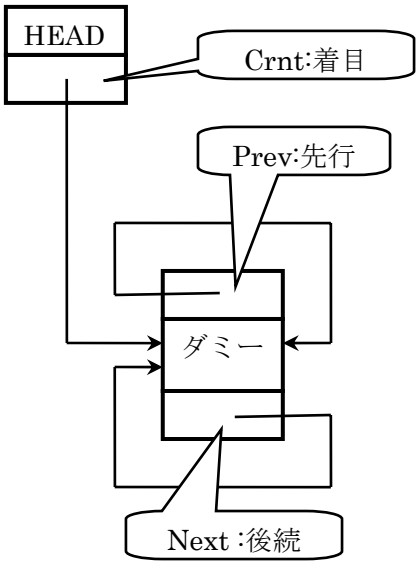

図 2. 循環・重複リストの初期化

まず、このプログラムでは、空のダミーノードを作成しを動的に生成し、リストの初期化を行う。先頭ポインタ(head)と着目ポインタ(crnt)がダミーノード自身を参照するようになっている。次に、ダミーノードを先行ポインタ(previous)と後続ポインタ(next)が、自己分自身のノードを参照するようになる。ダミーノードには値を格納しない。そのため、ノードがない場合でも、ダミーノードだけは存在する。

ダミーノードを作成し、ログの読込みが始まると同時に、ノードを追加しリストの生成を行う。ログオンとログオフの判別は、やはり pid で行う。3 章で述べたように pid は、ログオンもしくはログオフしたときに割り当てられる通し番号である。そのため、pid の大小を比較し、昇順に並べ替えれば、ログオンの情報を時系列に並べ替えることができる。

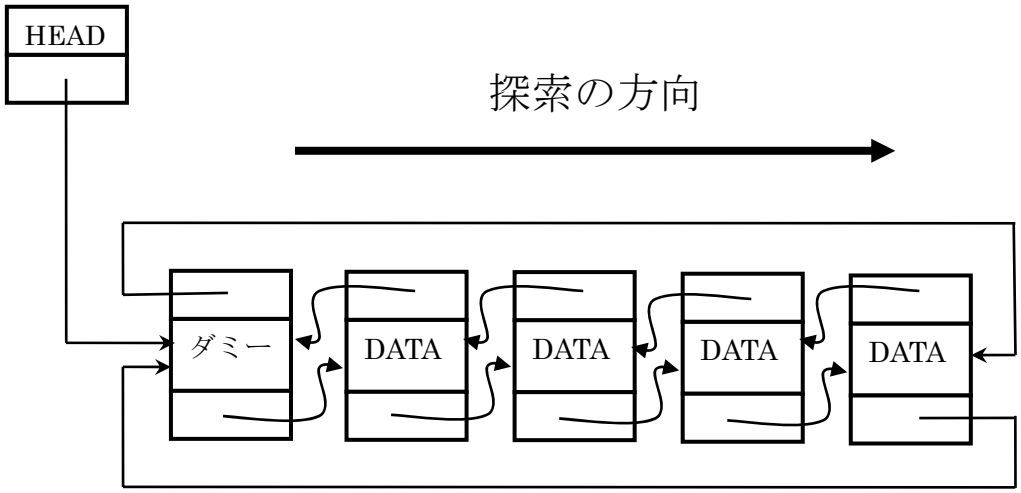

図 3. 循環・重複リストによる探索

ログオンとログオフは、線形索木で pid の値の大小を比較しながら行う。このとき、目的とする値をもつノードに出会うまで、順に探索を行う。探索は、ダミーノードには値がはいっていないので、ダミーノードの次のノードから開始する。探索が始まり、pid の値が一致すれば、ノードのデータ部分の変数”logonoff”に 0 を割り当てログオンとして処理する。そして、ログオンとして処理されたデータは、csv 形式で昇順に出力される。

4.2 使い方

このプログラムを利用する際には、実行ファイルと入力データであるログを同じディレクトリに保存しておかなければならない。また、読み込むファイルの名前をあらかじめ知っておく必要があるため、ファイル名が変更された場合は、そのたびにコンパイルする必要がある。

５．まとめ

5.1 結果と考察

プログラムについては、満足のいくものは作り出せなかった。もともとは、グラフィカルに利用状況を表示できるプログラムを考えていたが、その段階にまでは到達できなかった。しかし、データから必要な情報を抽出し、以前利用されていた利用状況表示システムの入力データとして利用できる形に近づけたことは、進展であると思う。

5.2 今後の課題

現在はプログラムが入っているディレクトリにデータが入ってきたことを監視するソフトを利用しなければ、自動的に起動することができない。そのため、自動的にファイルを読込み、時間ごとに起動する機能そして、読み込んだファイルを削除する機能を追加したい。それらが出来るようになれば、利用状況の表示を容易に行うことができると思う。

今回はログを読み込んでログオンの情報を取得するという方法だったが、DomainView*3のように自動的にログオン情報を取得し、利用状況をグラフィカルに表示できれば、情報システムの入れ替えにも対応することができると考えられる。

謝辞

この論文を作成するにあたり、多くの方の力をお借りしました。コーディングで行き詰まったときに相談にのってくれた堀田敬介先生。ログの提供やスクリプト作成を快く引き受けてくれた情報処理課の吉野さん。意見を出してくれたゼミ生、卒業生のみなさん。そして、アイディアやアドバイス、なかなか進まない研究を見守ってくれた根本先生。この研究に協力してくださったみなさん。本当にありがとうございました。

参考文献

[1] 柴田望洋，辻　良助「C 言語によるアルゴリズムとデータ構造」，ソフトバンクパブリッシング株式会社 （2002）

[2] 山地秀美，「C 言語文字列操作＋ファイル入出力完全制覇」，技術評論社 （2002）

[3] IT 用語辞典　e-Words　http://e-words.jp/

[4] ソフトウェア工房　WackyFactory　http://www.wackyfactory.net/domvw/

注釈

*1 AvtiveDirectory：ネットワーク上に存在する、サーバ、クライアント、プリンタなどの資源や、それらを利用するユーザやアクセス権などを一元管理する機能を持つソフトウェア。[3]

*2 Samba：UNIXで、ネットワーク上にあるwindowsマシンに、ファイル共有やプリンタ共有などのサービスを実現することができるソフトウェア。[3]

*3 DomainView：Microsoft Windows Networkに接続されたコンピュータの情報を高速に取得するフリーのソフトウェア。[4]

付録

ここには、4 章で説明したログを分析し、ログオンとログオフを判別させるプログラムのソースコードを載せた。主な機能は、ログを読込み、ログからログオンの判別に必要な文字列を切り出し、構造体に格納。そして、その構造体を pid でソーティングし判別し、ログオンとログオフを判別させている。結果は、csv 形式で主力している。

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


#define MAX 256

/* ― データ ―*/
typedef struct {
        int pid;            // pid
        int hour;           // 時
        int min;            // 分
        int sec;            // 秒
        int logonoff;               // ログオン(0)かログアウト(1)かの区別
        char room[10];              // コンピュータ教室名
        char machine[10];           // マシン番号
        char user[10];              //ユーザＩＤ
} Data;

/* ― ノード ― */
typedef struct node {
        Data data;                  // データ
        struct node *prev;  // 先行ノードへのポインタ
        struct node *next;  // 後続ノードへのポインタ
} Node;

/* ― 先頭ノードと着目ノードへのポインタ ― */
typedef struct {
```

```
        Node *head;         // 先頭ノードへのポインタ
        Node *crnt;         // 着目ノードへのポインタ
} List;

Node *talloc(void) {
        return ( (Node *)calloc(1, sizeof(Node)) );
}

/* ― リストの初期化 ― */
void InitList(List *list) {
        Node *dummy = talloc();     // ダミーノード生成
        list->head = list->crnt = dummy;
        dummy->prev = dummy->next = dummy;
}

void SetNode(Node *n, Data x, Node *prev, Node *next) {
        n->data = x;                // データの設定
        n->prev = prev;             // 先行ポインタの設定
        n->next = next;             // 後続ポインタの設定
}

/* ― リストが空かどうかを判定 ― */
int isEmptyList(List *list) {
        return ( (list->head)->next == list->head );
}

/* ― ノードの内容を表示 ― */
void PrintData(Data x) {
        printf("user:%s,machine:%s,time:%2d:%2d:%2d,pid:%d,Logon:%d¥n",x.user, x.machine,
x.hour, x.min, x.sec,x.pid,x.logonoff);
}

void PrintList(List *list) {
        FILE *out_file;

        out_file = fopen("list.csv","w");
```

```
	if(out_file == NULL){
		printf("ファイルオープン失敗¥n");
	}

	if ( isEmptyList(list) ) {
		printf("ノードがない¥n");
	}else{
		Node *ptr = (list->head)->next;		// dummy ノードの次から探索
			fprintf(out_file,"userID,machine,hour:min:sec,pid,logon¥n");
		while(ptr != list->head) {
			PrintData(ptr->data);

	fprintf(out_file,"%s,%s,%d:%d:%d,%d¥n",ptr->data.user,ptr->data.machine,ptr->data.hour,
ptr->data.min,ptr->data.pid);
			ptr = ptr->next;// 次のノードへ移動
			}
	}
	fclose(out_file);
}

/* — pが指すノードの後ろに挿入 — */
void InsertAfter(List *list, Node *p, Data x) {
	Node *ptr = talloc();		// メモリ確保
	Node *post = p->next;
	p->next = (p->next)->prev = ptr;
	SetNode(ptr, x, p, post);
	list->crnt = ptr;
}

/* — リストの先頭にノードを挿入 — */
void InsertFront(List *list, Data x) {
	InsertAfter(list, list->head, x);
}

/* — リストの末尾にノードを挿入 — */
void InsertRear(List *list, Data x) {
```

```
        InsertAfter(list, (list->head)->prev, x);
}

/* — pが指すノードを削除 — */
void RemoveNode(List *list, Node *p) {
        (p->prev)->next = p->next;
        (p->next)->prev = p->prev;
        free(p);
        list->crnt = p->prev;
}

/* — リストの先頭ノードを削除 — */
void RemoveFront(List *list) {
        if ( !isEmptyList(list) )
                RemoveNode(list, (list->head)->next);
}

/* — リストの末尾ノードを削除 — */
void RemoveRear(List *list) {
        if ( !isEmptyList(list) )
                RemoveNode(list, (list->head)->prev);
}

/* — リストの着目ノードを削除 — */
void Removecrnt(List *list) {
        if ( !isEmptyList(list) )
                RemoveNode(list, list->crnt);
}

/* — リストの全ノードを削除 — */
void ClearList(List *list) {
        while ( !isEmptyList(list) )
                RemoveFront(list);
}

/* — ノードのData の PID の大小比較 — */
```

```
int ChkPID(Data x, Data y) {
        return ( x.pid > y.pid );
}

Node *SearchNode(List *list, Data x) {
        Node *ptr = (list->head)->next;

        while (ptr != list->head) {
                if ( ChkPID(x, ptr->data) ) {
                        ptr = ptr->next;
                } else {
                        list->crnt = ptr->prev;
                        return (list->crnt);
                }
        }
        return (NULL);
}

void Read(List *list, char *fname) {
        FILE *infile;
        char line1[MAX], line2[MAX];
        char *p, *tmp;
        Data x;
        infile = fopen(fname, "r");
        if(infile == NULL){
                printf("ファイルの読み込み失敗¥n");
                exit(1);
        }
        //2行づつ読み込んでループ処理
        while( fgets(line1, MAX, infile_ptr) != NULL ) {      // 1行目読み込み
                fgets(line2, MAX, infile_ptr);                // 2行目読み込み
                p = strstr(line2, "(pid");          // 2行目にpidがあるか確認
                if ( p != NULL ) {
                        tmp = strtok(line2, " ");         //   1個目がマシン番号
                        strcpy(x.machine, tmp);           //   マシン番号
                        tmp = strtok(NULL, " ");          //   2個目がIPアドレス
```

```
                        tmp = strtok(NULL, " ");        //  3個目が connect
                        tmp = strtok(NULL, " ");        //  4個目が to
                        tmp = strtok(NULL, " ");        //  5個目が service
                        tmp = strtok(NULL, " ");        //  6個目が netlogon
                        tmp = strtok(NULL, " ");        //  7個目が initially
                        tmp = strtok(NULL, " ");        //  8個目が as
                        tmp = strtok(NULL, " ");        //  9個目が user
                        tmp = strtok(NULL, " ");        // 10個目が ユーザーID
                        strcpy(x.user, tmp);
                        tmp = strtok(NULL, " ");        // 11個目が  (uid=21496,
                        tmp = strtok(NULL, " ");        // 12個目が gid=107)
                        tmp = strtok(NULL, " ");        // 13個目が (pid
                        tmp = strtok(NULL, " ");        // 13個目が pid
                        x.pid = atoi(tmp);   // pid
                        tmp = strtok(line1, " ");       // 1個目が日付
                        tmp = strtok(NULL, " ");        // 2個目が時刻
                        x.hour = atoi( strtok(tmp, ":") );
                        x.min = atoi( strtok(NULL, ":") );
                        x.sec = atoi( strtok(tmp, ",") );
                        x.logonoff = 1;       // ログオン=0, ログオフ=1 初期化時は0

                        if ( isEmptyList(list) ) {
                                InsertFront(list, x);
                        } else {
        if ( SearchNode(list, x) != NULL ) {        // PIDを探索
        InsertAfter(list, list->crnt, x);  // NULL でなければ見つかった場所の次に挿入
                        } else {
                                InsertRear(list, x);  // NULL ならば末尾に挿入
                                }
                        }
                }
        }

        fclose(infile);
}
```

```
int main(void) {
        List list;

        InitList(&list);        // リストの初期化

        Read(&list, "ログのファイル名");//データの読込みとリストの生成

        PrintList(&list);       // リストの表示
        ClearList(&list);       // リストの後始末

        return(0);
}
```