SURVEY METHODOLOGY

SURVEY METHODOLOGY
This is the Subtitle

Robert M. Groves
Universitat de les Illes Balears

Floyd J. Fowler, Jr.
University of New Mexico

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Survey Methodology / Robert M. Groves . . . [et al.].
p. cm.—(Wiley series in survey methodology)
"Wiley-Interscience."
Includes bibliographical references and index.
ISBN 0-471-48348-6 (pbk.)
1. Surveys—Methodology. 2. Social sciences—Research—Statistical methods. I. Groves, Robert M. II. Series.

HA31.2.S873 2007
001.4'33—dc22 2004044064
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To my parents

CONTRIBUTORS

MASAYKI ABE, Fujitsu Laboratories Ltd., Fujitsu Limited, Atsugi, Japan

L. A. AKERS, Center for Solid State Electronics Research, Arizona State University, Tempe, Arizona

G. H. BERNSTEIN, Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, South Bend, Indiana; formerly of Center for Solid State Electronics Research, Arizona State University, Tempe, Arizona

CONTENTS IN BRIEF

CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

This is the foreword to the book.

PREFACE

This is an example preface. This is an example preface. This is an example preface. This is an example preface.

R. K. WATTS

Durham, North Carolina
September, 2007

ACKNOWLEDGMENTS

From Dr. Jay Young, consultant from Silver Spring, Maryland, I received the initial push to even consider writing this book. Jay was a constant “peer reader” and very welcome advisor durying this year-long process.

To all these wonderful people I owe a deep sense of gratitude especially now that this project has been completed.

G. T. S.

ACRONYMS

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association

GLOSSARY

NormGibbs	Draw a sample from a posterior distribution of data with an unknown mean and variance using Gibbs sampling.
pNull	Test a one sided hypothesis from a numberically specified posterior CDF or from a sample from the posterior
sintegral	A numerical integration using Simpson's rule

SYMBOLS

A Amplitude
& Propositional logic symbol
a Filter Coefficient

$\mathcal{B}$ Number of Beats

INTRODUCTION

CATHERINE CLARK, PHD.

Harvard School of Public Health
Boston, MA, USA

The era of modern began in 1958 with the invention of the integrated circuit by J. S. Kilby of Texas Instruments [**?**]. His first chip is shown in Fig. I. For comparison, Fig. I.2 shows a modern microprocessor chip, [**?**].

This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction.

$$ABCDEF\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

REFERENCES

1. J. S. Kilby, "Invention of the Integrated Circuit," *IEEE Trans. Electron Devices,* **ED-23,** 648 (1976).
2. R. W. Hamming, *Numerical Methods for Scientists and Engineers*, Chapter N-1, McGraw-Hill, New York, 1962.
3. J. Lee, K. Mayaram, and C. Hu, "A Theoretical Study of Gate/Drain Offset in LDD MOSFETs" *IEEE Electron Device Lett.,* **EDL-7**(3). 152 (1986).

PART I

SUBMICRON SEMICONDUCTOR MANUFACTURE

CHAPTER 1

THE SUBMICROMETER SILICON MOSFET

The sheer volumne of answers can often stifle insight...The purpose of computing is insight, not numbers.

—Hamming [**?**]

1.1 Here is a normal section

Here is some text.

1.1.1 This is the subsection

Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text.

1.1.1.1 This is the subsubsection

Here is some text after the subsubsection. Here is some text after the subsubsection. Here is some text after the subsubsection. Here is some text after the subsubsection.

This is the paragraph Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text.

1.2 Tips On Special Section Heads

Here are some things you can do for a special section head.

1.3 Break Long Section heads with double backslash

Here is some normal text. Here is some normal text. Here is some normal text.

1.4 Here is a Section Title

See this section head for information on how to explicitly break lines in table of contents.

1.5 How to get lower case in section head: pH

Here is some normal text. Here is some normal text. Here is some normal text.

1.6 How to use a macro that has both upper and lower case parts: V_{Txyz}

See the top of this file where the definition and box were set.

1.7 Equation

For optimal vertical spacing, no blank lines before or after equations

$$\alpha\beta\Gamma\Delta \tag{1.1}$$

as you see here.

CHAPTER 2

FIRST EDITED BOOK SAMPLE CHAPTER TITLE

G. Alvarez and R. K. Watts

Carnegie Mellon University, Pittsburgh, Pennsylvania

2.1 Here is a normal section

Here is some text.

Survey Methodology, Second Edition.

CHAPTER 3

SECOND EDITED BOOK SAMPLE CHAPTER TITLE

GEORGE SMEAL, PH.D.[1], SALLY SMITH, M.D.[2] AND STANLEY KUBRICK[1]

[1]AT&T Bell Laboratories Murray Hill, New Jersey
[2]Harvard Medical School, Boston, Massachusetts

3.1 Sample Section

Here is some sample text.

Survey Methodology, Second Edition.

3.2 Example, Figure and Tables

EXAMPLE 3.1 Optional Example Name

Use Black's law [Equation (6.3)] to estimate the reduction in useful product life if a metal line is initially run at 55°C at a maximum line current density.

illustration here

Figure 3.1 Short figure caption.

Figure 3.2 Oscillograph for memory address access operations, showing 500 ps address access time and superimposed signals of address access in 1 kbit memory plane.

Table 3.1 Small Table

one	two	three	four
C	D	E	F

Table 3.2 Effects of the two types of $\alpha\beta \sum_B^A$ scaling proposed by Dennard and co-workers[a,b]

Parameter	κ Scaling	κ, λ Scaling
Dimension	κ^{-1}	λ^{-1}
Voltage	κ^{-1}	κ^{-1}
Currant	κ^{-1}	λ/κ^2
Dopant Concentration	κ	λ^2/κ

[a] Refs. 19 and 20.

[b] $\kappa, \lambda > 1$.

3.2.1 Side by Side Tables and Figures

Space for figure...

Figure 3.3 This caption will go on the left side of the page. It is the initial caption of two side-by-side captions.

Space for second figure...

Figure 3.4 This caption will go on the right side of the page. It is the second of two side-by-side captions.

The command `\sidebyside{}{}` works similarly for tables:

Table 3.3 Table Caption

one	two	three	four
a	little	sample	table

Table 3.4 Table Caption

A	B	C	D
a	second little	sample	table

When using `\sidebyside`, one must use the cross referencing command `\label{}` after and *outside* of `\caption{}`:

```
\begin{table}
\sidebyside{\caption{Table Caption}\label{tab1}
first table}
{\caption{Table Caption}\label{tab2} second table}
\end{table}
```

or,

```
\begin{figure}
\sidebyside{\vskip<dimen>\caption{fig caption}\label{fig1}}
{\vskip<dimen>\caption{fig caption}\label{fig2}}
\end{figure}
```

3.3 Algorithm

This is a sample algorithm.

Algorithm 3.1

state_transition algorithm {
for each neuron $j \in \{0, 1, \ldots, M-1\}$
{
calculate the weighted sum S_j using Eq. (6);
if ($S_j > t_j$)
{turn ON neuron; $Y_1 = +1$}
else if ($S_j < t_j$)
{turn OFF neuron; $Y_1 = -1$}
else
{no change in neuron state; y_j remains unchanged;}
}
}

Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text. Here is some normal text.

> This is a sample of extract or quotation. This is a sample of extract or quotation. This is a sample of extract or quotation.

1. This is the first item in the numbered list.
2. This is the second item in the numbered list. This is the second item in the numbered list. This is the second item in the numbered list.

- This is the first item in the itemized list.
- This is the first item in the itemized list. This is the first item in the itemized list. This is the first item in the itemized list.

  This is the first item in the itemized list.

  This is the first item in the itemized list. This is the first item in the itemized list. This is the first item in the itemized list.

PROBLEMS

3.1 For Hooker's data, Problem 1.2, use the Box and Cox and Atkinson procedures to determine a appropriate transformation of PRES in the regression of PRES on TEMP. find $\hat{\lambda}$, $\tilde{\lambda}$, the score test, and the added variable plot for the score. Summarize the results.

3.2 The following data were collected in a study of the effect of dissolved sulfur on the surface tension of liquid copper (Baes and Killogg, 1953).

x = Weight % sulfur	Y = Decrease in Surface Tension (dynes/cm), two Replicates	
0. 034	301	316
0. 093	430	422
0. 30	593	586

a) Find the transformations of X and Y sot that in the transformed scale the regression is linear.

b) Assuming that X is transformed to $\ln(X)$, which choice of Y gives better results, Y or $\ln(Y)$? (Sclove, 1972).

c) In the case of α_1?

d) In the case of α_2?

3.3 Examine the Longley data, Problem 3.3, for applicability of assumptions of the linear model.

3.4 In the case of Γ_1?

3.5 In the case of Γ_2?

EXERCISES

3.1 For Hooker's data, Exercise 1.2, use the Box and Cox and Atkinson procedures to determine a appropriate transformation of PRES in the regression of PRES on

TEMP. find $\hat{\lambda}$, $\tilde{\lambda}$, the score test, and the added variable plot for the score. Summarize the results.

3.2 The following data were collected in a study of the effect of dissolved sulfur on the surface tension of liquid copper (Baes and Killogg, 1953).

x = Weight % sulfur	Y = Decrease in Surface Tension (dynes/cm), two Replicates	
0. 034	301	316
0. 093	430	422
0. 30	593	586

a) Find the transformations of X and Y sot that in the transformed scale the regression is linear.

b) Assuming that X is transformed to $\ln(X)$, which choice of Y gives better results, Y or $\ln(Y)$? (Sclove, 1972).

c) In the case of Δ_1?

d) In the case of Δ_2?

3.3 Examine the Longley data, Problem 3.3, for applicability of assumptions of the linear model.

3.4 In the case of Γ_1?

3.5 In the case of Γ_2?

3.4 Summary

This is a summary of this chapter. Here are some references: [1], [4].

REFERENCES

1. J. S. Kilby, "Invention of the Integrated Circuit," *IEEE Trans. Electron Devices,* **ED-23,** 648 (1976).
2. R. W. Hamming, *Numerical Methods for Scientists and Engineers*, Chapter N-1, McGraw-Hill, New York, 1962.
3. J. Lee, K. Mayaram, and C. Hu, "A Theoretical Study of Gate/Drain Offset in LDD MOSFETs" *IEEE Electron Device Lett.,* **EDL-7**(3). 152 (1986).
4. A. Berenbaum, B. W. Colbry, D.R. Ditzel, R. D Freeman, and K.J. O'Connor, "A Pipelined 32b Microprocessor with 13 kb of Cache Memory," it Int. Solid State Circuit Conf., Dig. Tech. Pap., p. 34 (1987).

Appendix: This is the Chapter Appendix Title

This is an appendix with a title.

$$\alpha\beta\Gamma\Delta \tag{A.1}$$

Figure 3-A.1 This is an appendix figure caption.

Table 3-A.1 This is an appendix table caption

Date	Event
1867	Maxwell speculated the existence of electromagnetic waves.
1887	Hertz showed the existence of electromagnetic waves.
1890	Branly developed technique for detecting radio waves.
1896	Marconi demonstrated wireless telegraph.
1897	Marconi patented wireless telegraph.
1898	Marconi awarded patent for tuned communication.
1898	Wireless telegraphic connection between England and France established.

Appendix

This is a Chapter Appendix without a title.

Here is a math test to show the difference between using Computer Modern math fonts and MathTimes math fonts. When MathTimes math fonts are used the letters in an equation will match TimesRoman italic in the text. (*g, i, y, x, P, F, n, f, etc.*) Caligraphic fonts, used for $\mathcal{ABC}$ below, will stay the same in either case.

$$g_i(y|f) = \sum_x P(x|F_n) f_i(y|x) \mathcal{ABC} \tag{B.1}$$

where $g_i(y|F_n)$ is the function specifying the probability an object will display a value y on a dimension i given F_n the observed feature structure of all the objects.

CHAPTER 4

HOME

CHAPTER 5

BASIC CONCEPT

CHAPTER 6

ENVIRONTMENT SETUP

CHAPTER 7

LIFE CYCLE

CHAPTER 8

CREATE OPERATION

CHAPTER 9

CLONE OPERATION

CHAPTER 10

PERFORM CHANGES

CHAPTER 11

REVIEW CHANGES

CHAPTER 12

COMMIT CAHNGES

CHAPTER 13

PUSH OPERATION

CHAPTER 14

UPDATE OPERATION

CHAPTER 15

STASH OPERATION

CHAPTER 16

MOVE OPERATION

Seperti namanya, operasi memindahkan direktori atau file dari satu lokasi ke lokasi lain. direktori yang dimodifikasi akan muncul sebagai berikut:
[tom@CentOS project] $ pwd

/home/tom/project
[tom@CentOS project] $ ls

README string string.c
[tom@CentOS project] $ mkdir src

[tom@CentOS project] $ git mv string.c src/

[tom@CentOS project] $ git status -s

R string.c −¿ src/string.c
?? string

Untuk membuat perubahan ini permanen, harus mendorong struktur direktori yang dimodifikasi ke repositori jauh sehingga pengembang lain dapat melihat ini.
[tom@CentOS project] $ git commit -m ”Modified directory structure”

[master 7d9ea97] Modified directory structure
1 files changed, 0 insertions(+), 0 deletions(-)
rename string.c =¿ src/string.c (100 %)

[tom@CentOS project] $ git push origin master
Counting objects: 4, done.
Compressing objects: 100 % (2/2), done.
Writing objects: 100 % (3/3), 320 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
e86f062..7d9ea97 master −¿ master

Di gudang lokal Jerry, sebelum operasi penarikan, ia akan menunjukkan struktur direktori lama.

[jerry@CentOS project] $ pwd
/home/jerry/jerry _repo/project

[jerry@CentOS project] $ ls
README string string.c

Tapi setelah operasi tarik, struktur direktori akan diperbarui. Sekarang, Jerry bisa melihat direktori src dan file yang ada di dalam direktori itu.
[jerry@CentOS project] $ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100 % (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100 % (3/3), done.
From git.server.com:project
e86f062..7d9ea97 master −¿ origin/master
First, rewinding head to replay your work on top of it...
Fast-forwarded master to 7d9ea97683da90bcdb87c28ec9b4f64160673c8a.

[jerry@CentOS project] $ ls
README src string

[jerry@CentOS project] $ ls src/
string.c

13.1 Hampir Semua Operasi Dilakukan Secara Lokal

Kebanyakan operasi pada Git hanya membutuhkan berkas-berkas dan resource lokal tidak ada informasi yang dibutuhkan dari komputer lain pada jaringan. Jika terbiasa dengan VCS terpusat dimana kebanyakan operasi memiliki overhead latensi jaringan, aspek Git satu ini akan membuat berpikir bahwa para dewa kecepatan telah memberkati Git dengan kekuatan. Karena memiliki seluruh sejarah dari proyek di lokal disk, dengan kebanyakan operasi yang tampak hampir seketika.

Sebagai contoh, untuk melihat history dari proyek, Git tidak membutuhkan data histori dari server untuk kemudian menampilkannya untuk, namun secara sedarhana Git membaca historinya langsung dari basisdata lokal proyek tersebut. Ini berarti melihat histori proyek hampir secara instant. Jika ingin membandingkan perubahan pada sebuah berkas antara versi saat ini dengan versi sebulan yang lalu, Git dapat mencari berkas yang sama pada sebulan yang lalu dan melakukan pembandingan perubahan secara lokal, bukan dengan cara meminta remote server melakukannya atau meminta server mengirimkan berkas versi yang lebih lama kemudian membandingkannya secara lokal.

Hal ini berarti bahwa sangat sedikit yang tidak bisa anda kerjakan jika sedang offline atau berada diluar VPN. Jika sedang berada dalam pesawat terbang atau sebuah kereta dan ingin melakukan pekerjaan kecil, dapat melakukan commit sampai anda memperoleh koneksi internet hingga anda dapat menguploadnya. Jika pulang ke rumah dan VPN client tidak bekerja dengan benar, tetap dapat bekerja. Pada kebanyakan sistem lainnya, melakukan hal ini cukup sulit atau bahkan tidak mungkin sama sekali. Pada Perforce misalnya, tidak dapat berbuat banyak ketika tidak terhubung dengan server; pada Subversion dan CVS, dapat mengubah berkas, tapi tidak dapat melakukan commit pada basisdata (karena tidak terhubung dengan basisdata). Hal ini mungkin saja bukanlah masalah yang besar, namun akan terkejut dengan perbedaan besar yang disebabkannya.

13.2 Segala sesuatu pada Git akan melalui proses checksum terlebih dahulu sebelum disimpan yang kemudian direferensikan oleh hasil checksum tersebut. Hal ini berarti tidak mungkin melakukan perubahan terhadap berkas manapun tanpa diketahui oleh Git. Fungsionalitas ini dimiliki oleh Git pada level terendahnya dan ini merupakan bagian tak terpisahkan dari filosofi Git. Tidak akan kehilangan informasi atau mendapatkan file yang cacat tanpa diketahui oleh Git.

Mekanisme checksum yang digunakan oleh Git adalah SHA-1 hash. Ini merupakan sebuah susunan string yang terdiri dari 40 karakter heksadesimal (0 hingga 9 dan a hingga f) dan dihitung berdasarkan isi dari sebuah berkas atau struktur direktori pada Git. sebuah hash SHA-1 berupa seperti berikut:
24b9da6552252987aa493b52f8696cd6d3b00373

Nilai seperti ini pada berbagai tempat di Git. Faktanya, Git tidak menyimpan nama berkas pada basisdatanya, melainkan nilai hash dari isi berkas.

Ketika melakukan operasi pada Git, kebanyakan dari operasi tersebut hanya menambahkan data pada basisdata Git. Seperti pada berbagai VCS, dapat kehilangan atau mengacaukan perubahan yang belum di-commit; namun jika anda melakukan com-

mit pada Git akan sangat sulit kehilangannya, terutama jika secara teratur melakukan push basisdata pada repositori lain.

Hal ini menjadikan Git menyenangkan karena kita dapat berexperimen tanpa kehawatiran untuk mengacaukan proyek. Git memiliki 3 keadaan utama dimana berkas anda dapat berada: committed, modified dan staged. Committed berarti data telah tersimpan secara aman pada basisdata lokal. Modified berarti telah melakukan perubahan pada berkas namun belum melakukan commit pada basisdata. Staged berarti telah menandai berkas yang telah diubah pada versi yang sedang berlangsung untuk kemudian dilakukan commit.

Direktori Git adalah dimana Git menyimpan metadata dan database objek untuk projek. Ini adalah bahagian terpenting dari Git, dan inilah yang disalin ketika anda melakukan kloning sebuah repository dari komputer lain.
Direktori kerja adalah sebuah checkout tunggal dari satu versi dari projek. Berkas-berkas ini kemudian ditarik keluar dari basisdata yang terkompresi dalam direktori Git dan disimpan pada disk untuk anda gunakan atau modifikasi.

Staging area adalah sebuah berkas sederhana, umumnya berada dalam direktori Git, yang menyimpan informasi mengenai apa yang menjadi commit selanjutnya. Ini terkadang disebut sebagai index, tetapi semakin menjadi standard untuk menyebutnya sebagai staging area.Alur kerja dasar Git adalah seperti ini:

Jika sebuah versi tertentu dari sebuah berkas telah ada di direktori git dianggap 'committed'. Jika berkas diubah (modified) tetapi sudah ditambahkan ke staging area maka itu adalah 'staged'. Dan jika berkas telah diubah sejak terakhir dilakukan checked out tetapi belum ditambahkan ke staging area maka itu adalah 'modified'.

13.3 Perintah Untuk Membuat Sebuah Proyek

Membuat direktori baru di repositori Git dengan git init. Melakukan direktori setiap saat, benar-benar lokal. Executive git init dalam direktori, Membuat Git repositori. Sebagai contoh, buat item w3big:

```
$ mkdir w3big
$ cd w3big/
$ git init
Initialized empty Git repository in /Users/tianqixin/www/w3big/.git/
# /www/w3big/.git/ Git
```

Sekarang dapat melihat subdirektori git yang dihasilkan dalam proyek. Ini adalah repositori Git, dan semua data yang terkait dengan snapshot dari proyek disimpan di sini.

```
ls -a
.               ..              .git
```

Gunakan git clone repositori Git untuk salinan lokal, sehingga dapat melihat item atau memodifikasinya. Jika membutuhkan sebuah proyek kerjasama dengan orang

lain atau ingin menyalin sebuah proyek, melihat kode, dapat mengkloning proyek. Jalankan:
git clone [url]

Sebagai contoh, kloning proyek pada Github:
$ git clone git@github.com:schacon/simplegit.git
Cloning into 'simplegit'...
remote: Counting objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Receiving objects: 100 % (13/13), done.
Resolving deltas: 100 % (2/2), done.
Checking connectivity... done.

Setelah kloning selesai di direktori saat ini akan menghasilkan simplegit direktori:
$ Cd simplegit / $ ls README Rakefile lib
operasi akan menyalin semua catatan proyek.

$ ls -a
. .. .git README Rakefile lib
$ cd .git
$ ls
HEAD description info packed-refs
branches hooks logs refs
config index objects
Secara default, Git akan mengikuti nama URL yang tersedia item untuk

membuat direktori proyek lokalditunjukkan. URL biasanya nama item terakhir / setelah. Jika ingin nama yang berbeda dapat menambahkan nama yang inginkan setelah perintah.

13.4 Snapshot Dasar

Pekerjaan Git adalah untuk membuat dan menyimpan snapshot dari proyek dan setelah snapshot dan membandingkan. Bab ini akan tentang menciptakan sebuah snapshot dari proyek dan mengirimkan pengenalan perintah.

git add perintah untuk menambahkan file ke cache, seperti yang tambahkan dua file berikut:
$ touch README
$ touch hello.php
$ ls
README hello.php

$ git status -s
?? README
?? hello.php
$
Perintah git status digunakan untuk melihat status proyek. Selanjutnya jalankan git add perintah untuk menambahkan file:
$ git add README hello.php
Sekarang jalankan git status, dapat melihat dua dokumen tersebut telah ditambahkan

untuk pergi.
$ git status -s
A README
A hello.php
$

Proyek baru, menambahkan semua file yang sama, kita dapat menggunakangit add. Perintah untuk menambahkan semua file dalam proyek saat ini. Sekarang memodifikasi file README:
$ vim README
¡pre¿
¡p¿ README Ł¡b¿ # w3big Git ¡/b¿¡/p¿
¡p¿ git status¡/p¿
$ git status -s
AM README
A hello.php

”AM” status berarti bahwa file tersebut setelah kami menambahkannya ke cache ada perubahan. Setelah perubahan menjalankan git add perintah untuk menambahkannya ke cache:
$ git add .
$ git status -s
A README
A hello.php
Bila ingin perubahan yang terkandung dalam snapshot laporan yang akan datang dalam waktu, harus menjalankan git add.

Git status untuk melihat setelah komit terakhir jika ada perubahan. Menunjukkan perintah ini ketika ditambahkan -s parameter untuk mendapatkan hasil yang singkat. Jika tidak menambahkan parameter ini akan keluaran rinci:
$ git status
On branch master
Initial commit

Changes to be committed:

(use ”git rm –cached ¡file¿...” to unstage)

new file: README

new file: hello.php

Status git diff git eksekutif untuk melihat rincian hasil eksekusi. Git perintah diff dan menampilkan cache write telah dimodifikasi tapi belum ditulis ke cache perubahan perbedaan. git diff Ada dua skenario utama.

- Perubahan tidak cache:diff git
- Lihatperubahan cache: git diff –cached
- Lihat cache dan uncached semuaperubahan: git diff KEPALA
- Tampilkan ringkasan daripada seluruhdiff: git diff –stat

Masukkan berikut dalam file hello.php:
¡?php
echo 'www.w3big.com';
?¿
$ git status -s
A README
AM hello.php
$ git diff
diff –git a/hello.php b/hello.php
index e69de29..69b5711 100644
— a/hello.php
+++ b/hello.php
@@ -0,0 +1,3 @@
+¡?php
+echo 'www.w3big.com';
+?¿
Menampilkan status git pada untuk berubah setelah update atau menulis garis pe-

rubahan cache dengan garis dan git diff menunjukkan secara spesifik apa perubahan tersebut. Selanjutnya melihat git berikutnya diff pelaksanaan –cached hasil:
$ git add hello.php
$ git status -s
A README
A hello.php
$ git diff –cached
diff –git a/README b/README
new file mode 100644
index 0000000..8f87495
— /dev/null
+++ b/README
@@ -0,0 +1 @@

+ # w3big Git
diff –git a/hello.php b/hello.php
new file mode 100644
index 0000000..69b5711
— /dev/null
+++ b/hello.php
@@ -0,0 +1,3 @@
+¡?php
+echo 'www.w3big.com';
+?¿

Gunakan git menambahkan perintah ingin menulis isi dari buffer snapshot, dan mengeksekusi git commit akan menambahkan konten ke gudang penyangga. Git mengirimkan masing-masing nama dan alamat e-mail yang tercatat, sehingga langkah pertama perlu mengkonfigurasi nama pengguna dan alamat e-mail.
$ git config –global user.name 'w3big'
$ git config –global user.email test@w3big.com
Berikutnya menulis caching, dan menyerahkan semua perubahan hello.php

tersebut. Dalam contoh pertama menggunakan opsi -m untuk memberikan baris perintah untuk mengirimkan komentar.
$ git add hello.php
$ git status -s
A README
A hello.php
$ $ git commit -m ''
[master (root-commit) d32cf1f]
2 files changed, 4 insertions(+)
create mode 100644 README
create mode 100644 hello.php

Sekarang telah mencatat snapshot. Jika kita jalankan git status:

$ git status
# On branch master
nothing to commit (working directory clean)

Output di atas menunjukkan bahwa setelah pengajuan terakhir, tidak membuat perubahan apapu. Jika tidak menetapkan opsi -m, Git mencoba untuk membuka editor untuk mengisi informasi yang disampaikan. Git jika tidak dapat menemukan informasi yang relevan dalam konfigurasi, default akan membuka vim. Layar akan terlihat seperti ini:
Please enter the commit message for your changes. Lines starting
with ' #' will be ignored, and an empty message aborts the commit.

On branch master
Changes to be committed:
(use ”git reset HEAD ¡file¿...” to unstage)
#
modified: hello.php
#
~
~
”.git/COMMIT _EDITMSG” 9L, 257C

Jika berpikir git add disampaikan proses cache yang terlalu rumit, Git juga memungkinkan Anda untuk menggunakan opsi -a untuk melewatkan langkah ini. format perintah adalah sebagai berikut:
git commit -a

Mari memodifikasi file hello.php sebagai berikut:
¡?php
echo ’www.w3big.com’;
echo ’www.w3big.com’;
?¿

Kemudian jalankan perintah berikut:
git commit -am ’ hello.php ’
[master 71ee2cb] hello.php
1 file changed, 1 insertion(+)

Git reset perintah HEAD untuk menghapus konten cache. Mari mengubah file berkas README, sebagai berikut:
w3big Git
#

File hello.php diubah sebagai berikut:
¡?php
echo ’www.w3big.com’;
echo ’www.w3big.com’;
echo ’www.w3big.com’;
?¿

Sekarang setelah dua file diubah disampaikan ke zona penyangga, sekarang ingin membatalkan salah satu dari cache, sebagai berikut:
$ git status -s
M README
M hello.php
$ git add .
$ git status -s

```
M  README
M  hello.pp
$ git reset HEAD – hello.php
Unstaged changes after reset:
M                hello.php
$ git status -s
M  README
M hello.php
```

Sekarang menjalankan git commit, perubahan hanya akan diserahkan berkas README, tapi hello.php tidak.

```
$ git commit -m ''
[master f50cfda]
1 file changed, 1 insertion(+)
$ git status -s
M hello.php
```

Melihat file perubahan hello.php dan untuk pengajuan. Maka dapat menggunakan perintah berikut untuk memodifikasi hello.php menyerahkan:

```
$ git commit -am ' hello.php '
[master 760f74d]  hello.php
1 file changed, 1 insertion(+)
$ git status
On branch master
nothing to commit, working directory clean
```

Singkatnya, melakukan git reset HEAD untuk membatalkan sebelum git add untuk menambahkan, tetapi tidak ingin untuk memasukkan dalam cache snapshot di commit selanjutnya.

Entri rm git akan dihapus dari cache. ulang KEPALA git ini membatalkan entri cache yang berbeda. "Batal Cache", yang berarti bahwa pemulihan akan membuat perubahan ke cache. Secara default, git file rm akan dihapus dari file cache dan hard drive (direktori kerja). Jika ingin menyimpan file dalam direktori kerja, dapat menggunakangit rm –cached: Seperti kita menghapus hello.php file:

```
$ git rm hello.php
rm 'hello.php'
$ ls
README
```

Tidak menghapus file dari ruang kerja:

```
$ git rm –cached README
rm 'README'
$ ls
README
```

Git perintah mv untuk melakukan semua hal yanggit rm perintah operasi –cached,mengubah nama file pada disk, dan kemudian jalankan git add untuk menambahkan file baru ke cache. README pertama kita hapus hanya menambahkan kembali:

```
$ git add README
```

Kemudian nama yang sama yaitu:

```
$ git mv README README.md
$ ls
README.md
```

CHAPTER 17

RENAME OPERATION

Sampai sekarang, baik Tom dan Jerry menggunakan perintah manual untuk menyusun proyek mereka. Sekarang, Jerry memutuskan untuk membuat Makefile untuk proyek mereka dan juga memberi nama yang tepat untuk file ”string.c”.
[jerry@CentOS project] $ pwd
/home/jerry/jerry _repo/project
[jerry@CentOS project] $ ls

README src
[jerry@CentOS project] $ cd src/

[jerry@CentOS src] $ git add Makefile

[jerry@CentOS src] $ git mv string.c string _operations.c

[jerry@CentOS src] $ git status -s

A Makefile
R string.c −¿ string _operations.c

Git menunjukkan R sebelum nama file untuk menunjukkan bahwa file telah diganti namanya.

Untuk komit operasi, Jerry menggunakan - bendera, yang membuat git komit secara otomatis mendeteksi file yang dimodifikasi.
[jerry@CentOS src] $ git commit -a -m 'Added Makefile and renamed strings.c to string _operations.c '

[master 94f7b26] Added Makefile and renamed strings.c to string _operations.c
1 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/Makefile
rename src/ {string.c =¿ string _operations.c } (100 %)

Setelah komit, dia mendorong perubahannya ke repositori.
[jerry@CentOS src] $ git push origin master

Perintah di atas akan menghasilkan hasil sebagai berikut:
Counting objects: 6, done.
Compressing objects: 100 % (3/3), done.
Writing objects: 100 % (4/4), 396 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
7d9ea97..94f7b26 master −¿ master

Sekarang, pengembang lain dapat melihat modifikasi ini dengan memperbarui repositori lokal mereka.

Kegunaan utama dari sistem kontrol versi ialah sebagai alat untuk manajemen kode program. Terdapat dua kegunaan utama dari sistem ini, yaitu:
Menggabungkan perubahan-perubahan kode dari versi lama (misal: untuk mengembalikan fitur yang telah dihapus) ataupun menggabungkan perubahan dari orang lain (misal: menggabungkan fitur yang dikembangkan oleh anggota tim lain).

14.1 Intalasi Git

git berjalan pada semua sistem operasi populer (Mac, Windows, Linux). Jika menggunakan Windows atau Mac, masuk ke situs utama git pada lalu lakukan download dan instalasi software tersebut. Pengguna Linux dapat melakukan instalasi melalui repositori distribusi yang dilakukan, melalui perintah sejenis:
yum install git

pada repositori berbasis RPM, atau perintah
apt-get install git

Untuk repositori berbasis deb. Kembali lagi, perintah hanya diberikan untuk distribusi paling populer (Debian / Ubuntu dan RedHat / Fedora), karena keterbatasan ruang. Jika menggunakan distrusi lain (seperti Gentoo atau Arch, maka diasumsikan telah mengetahui cara instalasi git atau perangkat lunak lain pada umumnya).

Khusus untuk sistem operasi Windows, pastikan instalasi anda diambil dari , karena pada paket yang tersedia di website tersebut telah diikutkan juga OpenSSH, yang akan sangat berguna jika ingin berkolaborasi dengan programmer lain. Perintah git juga harus memberikan respon yang benar:

```
bert@LYNNSLENIA ~
$ git
usage: git [–version] [–exec-path[=¡path¿]] [–html-path] [–man-path] [–info-path]
       [-p |–paginate |–no-pager] [–no-replace-objects] [–bare]
       [–git-dir=¡path¿] [–work-tree=¡path¿] [–namespace=¡name¿]
       [-c name=value] [–help]
       ¡command¿ [¡args¿]
The most commonly used git commands are:
  add       Add file contents to the index
  bisect    Find by binary search the change that introduced a bug
  branch    List, create, or delete branches
  checkout  Checkout a branch or paths to the working tree
  clone     Clone a repository into a new directory
  commit    Record changes to the repository
 diff      Show changes between commits, commit and working tree, etc
  fetch     Download objects and refs from another repository
  grep      Print lines matching a pattern
  init      Create an empty git repository or reinitialize an existing one
  log       Show commit logs
  merge     Join two or more development histories together
  mv        Move or rename a file, a directory, or a symlink
  pull      Fetch from and merge with another repository or a local branch
  push      Update remote refs along with associated objects
  rebase    Forward-port local commits to the updated upstream head
  reset     Reset current HEAD to the specified state
  rm        Remove files from the working tree and from the index
  show      Show various types of objects
  status    Show the working tree status
  tag       Create, list, delete or verify a tag object signed with GPG
 See 'git help ¡command¿' for more information on a specific command.

bert@LYNNSLENIA ~
$
```

14.2 Inisiasi

Untuk dapat menggunakan sistem kontrol versi, terlebih dahulu kita harus mempersiapkan repositori. Sebuah repositori menyimpan seluruh versi dari kode program kita. Tidak usah takut, karena repositori tidak akan memakan banyak ruang *hard disk*, karena penyimpanan tidak dilakukan terhadap keseluruhan file. Repositori hanya akan menyimpan *perubahan* yang terjadi pada kode kita dari satu versi ke

versi lainnya. Bahasa kerennya, repositori hanya menyimpan delta dari kode pada setiap versinya.

Pada (di saat kontrol versi yang populer adalah cvs dan programmer pada umumnya berjanggut putih), membangun repositori kode baru adalah hal yang sangat sulit dilakukan. Harus memiliki sebuah *server* khusus yang dapat diakses oleh seluruh anggota tim. Jika server tidak dapat diakses karena jaringan rusak atau internet putus, maka tidak dapat melakukan kontrol versi (dan harus kembali ke metode direktori, atau tidak bekerja).

git merupakan sistem kontrol versi terdistribusi, yang berarti git dapat dijalankan tanpa perlu adanya repositori terpusat. Yang diperlukan untuk membuat repositori ialah mengetikkan perintah tertentu di direktori utama. Mulai membuat repositori baru.:

```
bert@LYNNSLENIA ~
$ cd Desktop/projects/git-tutor/
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ ls
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$
```

Menambahkan kode baru ke dalam direktori ini. Buat sebuah file baru yang bernama

cerita.txt di dalam direktori tersebut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ echo ”ini adalah sebuah cerita” ¿ cerita.txt
 bert@LYNNSLENIA ~/Desktop/projects/git-tutor

$ ls
cerita.txt
```

kemudian masukkan perintah git init untuk melakukan inisialisasi repositori:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ git init
Initialized empty Git repository in c:/Users/bert/Desktop/projects/git-tutor/.git/
```

Setelah melakukan inisialisasi, git secara otomatis akan membuat direktori .git pada repositori (lihat potongan kode di bawah). Direktori tersebut merupakan direktori yang digunakan oleh git untuk menyimpan basis data delta kode, dan berbagai metadata lainnya. Mengubah direktori tersebut dapat menyebabkan hilangnya seluruh *history* dari kode.

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls -a
. .. .git cerita.txt
```

14.3 Penambahan File ke Repository

Penyimpanan sejarah dapat dimulai dari saat pertama: kapan file tersebut dibuat dan ditambahkan ke dalam repositori. Untuk menambahkan file ke dalam repositori, gunakan perintah git add:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git add .
warning: LF will be replaced by CRLF in cerita.txt.
The file will have its original line endings in your working directory.
```

Secara sederhana, sintaks dari perintah git add adalah sebagai berikut:

```
git add [nama file atau pola]
```

Memasukkan nama file dalam perintah git add pada dasarnya akan memerintahkan git untuk menambahkan **semua** file baru dalam repositori. Jika hanya ingin menambahkan satu file (misalkan ada file yang belum yakin akan ditambahkan ke repositori), nama file spesifik dapat dimasukkan:

```
git add cerita.txt
```

Setelah menambahkan file ke dalam repositori, harus melakukan *commit*. Perintah *commit* memberitahukan kepada git untuk menyimpan sejarah dari file yang telah ditambahkan. Pada git, penambahan, perubahan, ataupun penghapusan sebuah file baru akan tercatat jika perntah *commit* telah dijalankan. Mari lakukan *commit* dengan menjalankan perintah git commit:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
```

Jika langkah di atas diikuti dengan benar, maka kembali ke git bash, dengan pesan berikut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
[master (root-commit) 1d4cdc9] Inisialisasi repo. Penambahan cerita.txt.
warning: LF will be replaced by CRLF in cerita.txt.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+)
create mode 100644 cerita.txt
```

14.4 Mengubah Isi File

Kegunaan utama kontrol versi (yang tercermin dari namanya) ialah melakukan manajemen perubahan secara otomatis untuk kita. dan kemudian jalankan perintah git commit lagi:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
# On branch master
# Changes not staged for commit:
#   (use "git add ¡file¿..." to update what will be committed)
#   (use "git checkout – ¡file¿..." to discard changes in working directory)
```

```
#
#       modified:  cerita.txt
#
no changes added to commit (use ”git add” and/or ”git commit -a”)
```

Perhatikan bahwa git secara otomatis mengetahui file mana saja yang berubah, tetapi tidak melakukan pencatatan perubahan tersebut. Untuk memerintahkan git mencatat perubahan tersebut, gunakan perintah git commit -a:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit -a
[master 61c4707] Kapitalisasi dan melengkapi kalimat.
1 file changed, 1 insertion(+), 1 deletion(-)
```

Selain melakukan perubahan, tentunya terkadang kita ingin mengetahui perubahan-perubahan apa saja yang terjadi selama pengembangan. Untuk melihat daftar perubahan yang telah dilakukan, kita dapat menggunakan perintah git log:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git log
commit 61c47074ee583dbdd16fa9568019e80d864fb403
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date:   Sun Dec 23 16:36:46 2012 +0700
    Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698

Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date:   Sun Dec 23 16:10:33 2012 +0700
    Inisialisasi repo. Penambahan cerita.txt.
```

Mari jalankan perintah git log sekali lagi, untuk melihat hasil pekerjaan kita sejauh ini:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git log
commit 28dabb1c54a086cce567ecb890b10339416bcbfa
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date:   Sun Dec 23 16:49:21 2012 +0700
    Penambahan misteri terbesar di dunia.

commit 61c47074ee583dbdd16fa9568019e80d864fb403

Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date:   Sun Dec 23 16:36:46 2012 +0700
    Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698
```

Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date: Sun Dec 23 16:10:33 2012 +0700
Inisialisasi repo. Penambahan cerita.txt.

git memungkinkan kita untuk mengembalikan kode ke dalam keadaan sebelumnya, yaitu *commit* terakhir. Melakukan pengembalian kode ini dengan menggunakan perintah git checkout seperti berikut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout HEAD – cerita.txt
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dalam goa.
Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
```

Parameter HEAD pada perintah yang kita jalankan merupakan parameter untuk memberitahukan git checkout bahwa kita ingin mengembalikan kode pada revisi terakhir (HEAD dalam istilah git). Karena hanya ingin mengembalikan file cerita.txt, maka kita harus memberitahukan git checkout, melalui parameter – cerita.txt. Perintah git checkout juga memiliki banyak kegunaan lainnya selain mengembalikan kode ke revisi tertentu.

Untuk melihat bagaimana fitur ini bekerja, mari lakukan perubahan pada repositori terlebih dahulu. Tambahkan sebuah file baru ke dalam repositori:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt
 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)

$ echo ”Seekor kera, terpuruk, terpenjara dalam goa. Di gunung suci sunyi
tempat hukuman para dewa.” ¿ lagu-intro.txt
 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)

$ ls
cerita.txt  lagu-intro.txt
 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)

$ git add .
warning: LF will be replaced by CRLF in lagu-intro.txt.
The file will have its original line endings in your working directory.
 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
```

```
$ git commit
[master 03d0628] Penambahan lagu intro.
warning: LF will be replaced by CRLF in lagu-intro.txt.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+)
create mode 100644 lagu-intro.txt
```

Kemudian kita akan melakukan edit terhadap cerita.txt dan mengganti nama lagu-intro.txt menjadi lagu-intro-awal.txt:

```
ert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro.txt
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ notepad cerita.txt
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ mv lagu-intro.txt lagu-intro-awal.txt
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro-awal.txt
```

Setelah melakukan perubahan tersebut, kita mengalami amnesia sesaat karena kucing kantor jatuh ke kepala kita (kucing yang menyebalkan!). Karena telah lupa akan perubahan yang dilakukan, kita dapat melihat apa saja yang berubah dengan menggunakan perintah git status:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm ¡file¿..." to update what will be committed)
#   (use "git checkout – ¡file¿..." to discard changes in working directory)
#
#       modified:  cerita.txt
#       deleted:   lagu-intro.txt
#
# Untracked files:
#   (use "git add ¡file¿..." to include in what will be committed)
#
#       lagu-intro-awal.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

Perhatikan bahwa terdapat dua bagian dari status yang diberikan:
"Changes not staged for commit " menampilkan daftar file yang berubah, tetapi belum di-*commit*. File yang tercatat ini termasuk file yang diubah dan dihapus.
"Untracked files " menampilkan file yang belum ditambahkan ke dalam repositori.

Jika ingin melihat apa saja yang diubah pada file cerita.txt, kita dapat menggunakan perintah git diff:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git diff cerita.txt
diff –git a/cerita.txt b/cerita.txt
index 846114d..dbcb596 100644
— a/cerita.txt
+++ b/cerita.txt
@@ -1,3 +1,3 @@
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dala
 -Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?

\ No newline at end of file
+Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!
\ No newline at end of file
(END)
```

Format yang ditampilkan mungkin agak membingungkan, tetapi tidak usah takut, karena bagian yang perlu diperhatikan hanyalah pada bagian yang bertanda - dan +. Pada git bash, bahkan bagian ini diberi warna (merah untuk - dan hijau untuk +). Tanda +, tentunya berarti bagian yang ditambahkan, dan tanda - berarti bagian yang dihapus. Dengan melihat perubahan pada baris yang bersangkutan, kita dapat mengetahui bahwa ? diubah menjadi ???! pada akhir baris.

Setelah mengetahui perubahan yang dilakukan, dan menganggap perubahan tersebut aman untuk di-*commit*, kita lalu dapat melakukan *commit* seperti biasa:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git add lagu-intro-awal.txt
warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
The file will have its original line endings in your working directory.

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
[master 306f422] Dramatisasi cerita dan perubahan nama file lagu.
warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+)
create mode 100644 lagu-intro-awal.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git log
commit 306f42258f4bfee95d10396777391ae013bc6edd
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date:   Sun Dec 23 18:22:30 2012 +0700

    Dramatisasi cerita dan perubahan nama file lagu.

commit 03d06284462f7fc43b610d522678f4f22cdd9a40
```

Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date: Sun Dec 23 18:08:10 2012 +0700

Penambahan lagu intro.

commit 28dabb1c54a086cce567ecb890b10339416bcbfa
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date: Sun Dec 23 16:49:21 2012 +0700

Penambahan misteri terbesar di dunia.

commit 61c47074ee583dbdd16fa9568019e80d864fb403
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date: Sun Dec 23 16:36:46 2012 +0700

Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698
Author: Alex Xandra Albert Sim ¡bertzzie@gmail.com¿
Date: Sun Dec 23 16:10:33 2012 +0700

Inisialisasi repo. Penambahan cerita.txt.

14.5 Membaca File Lama, dan Menjalankan Mesin Waktu

Nomor revisi, seperti yang telah dijelaskan sebelumnya, berguna sebagai tanda untuk memisahkan antara satu *commit* dengan *commit* lainnya. Misalnya jika ingin melihat isi file cerita.txt pada saat awal pertama kali dibuat, kita dapat menggunakan perintah git show, yang sintaksnya adalah:

git show [nomor revisi]:[nama file]

contoh pengunaan:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git show 1d4cdc:cerita.txt
ini adalah sebuah cerita
```

Perhatikan bahwa nomor commit yang dimasukkan hanyalah enam karakter saja. Jika keenam karakter tersebut sama untuk beberapa nomor *commit*, kita baru perlu memasukkan karakter selanjutnya, sampai tidak terdapat konflik nama lagi.

Sesuai dengan nomor revisi dengan menggunakan git checkcout yang telah dijelaskan sebelumnya. Contohnya :

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro-awal.txt
```

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dalam
goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 61c470 cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dalam
goa.
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 1d4cdc cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
ini adalah sebuah cerita

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 03d0628 cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dalam
goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout HEAD cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara dalam
goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!
```

Perhatikan bahwa pada saat menggunakan perintah git checkout, menggunakan cat untuk melihat isi file. Hal ini dikarenakan git checkout benar-benar mengubah file yang ada pada repositori, berbeda dengan git show yang hanya menampilkan file tersebut pada revisi tertentu.

REFERENCES

[Kil76] J. S. Kilby, "Invention of the Integrated Circuit," *IEEE Trans. Electron Devices,* **ED-23,** 648 (1976).

[Ham62] R. W. Hamming, *Numerical Methods for Scientists and Engineers*, Chapter N-1, McGraw-Hill, New York, 1962.

[Hu86] J. Lee, K. Mayaram, and C. Hu, "A Theoretical Study of Gate/Drain Offset in LDD MOSFETs" *IEEE Electron Device Lett.,* **EDL-7**(3). 152 (1986).

[Ber87] A. Berenbaum, B. W. Colbry, D.R. Ditzel, R. D Freeman, and K.J. O'Connor, "A Pipelined 32b Microprocessor with 13 kb of Cache Memory," it Int. Solid State Circuit Conf., Dig. Tech. Pap., p. 34 (1987).