2 TIME-SHARING AND VIRTUALIZATION

Intimacies of the User: From the Stolen Look to Stolen Time

It is October 1972, and *Rolling Stone* is visiting Stanford University. Although accredited as a sports reporter, journalist Stewart Brand is not there to watch a traditional college sport; instead, he is there for the computer game Spacewar (figure 2.1). Ushered into the computer lab, Brand watches as long-haired programmers in front of a glowing TV-like tube grab their joysticks, maneuver their spaceships, and fire photon torpedoes, "joyously slaying their friend and wasting their employers' valuable computer time."[1]

Brand's description would have been unfathomable for a typical reader of the early 1970s. Not only were the programmers not using computers—office machinery that cost hundreds of thousands of dollars apiece—for work, they were using them interactively. The usual way of using computers at the time was called batch processing, in which users would submit punch cards or magnetic tape to an operator and receive the results in hours, even a few days later—a process Brand derided as "passive consumerism: data was something you sent to the manufacturer, like color film."[2] Instead, at the Stanford AI lab, a console would ask a programmer questions ("HOW MANY SPACE MINES DO YOU WANT?"), and he or she would type back an answer. Then, a fraction of a second later, the computer would draw the mines on the console—no punch cards involved.

For today's critics as much as Brand, Spacewar represented a historical turning point toward personal computing. In his 2003 book, for example, historian Paul Ceruzzi writes that "the way it was being used was personal: for fun, interactively, with no concern for how many ticks of the processor one was using."[3] Appropriately, the *Rolling Stone* article on Spacewar opens his chapter on the personal computer, for in Ceruzzi's reading of it, this moment

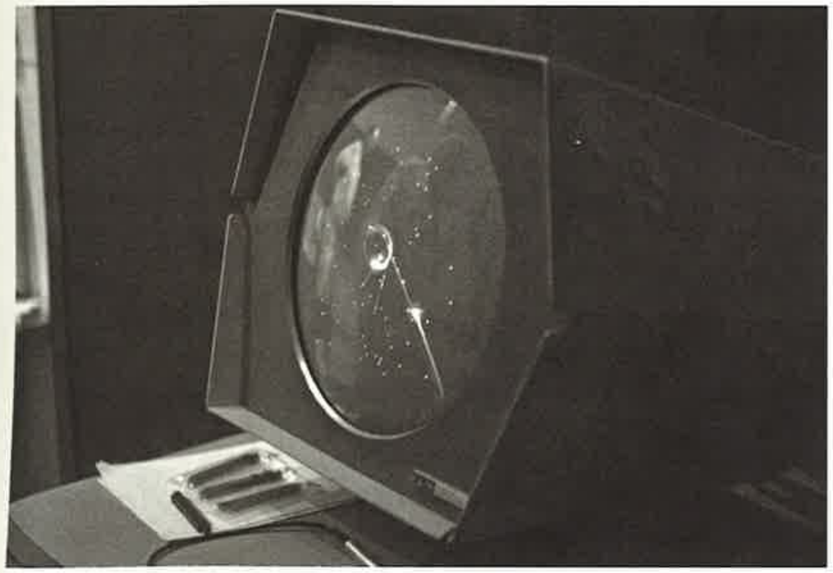

Figure 2.1

Spacewar! running on a PDP-1. (The Computer History Museum in Mountain View, California runs a demo of *Spacewar* every first and third Saturday.) © Joi Ito, reprinted under Creative Commons BY 2.0 license.

would offer the first "mental model" of a user of personal computers. This model meant that computing could be used for 'personal' matters, such as dating, gaming, and so on. But personal computing would also mean something that may seem odd to a contemporary reader: the illusion that a person would be "given the full attention and resources of the computer."[4]

We might find that statement a tautology: of course each person has the full 'attention and resources' of his or her own computer! At the time of Brand's article, however, only a single computer served the dozens of programmers at the Stanford lab, a PDP-10 next door to the room where the competitors were playing Spacewar. It was far too costly for any single individual to own a computer—but new software known as time-sharing made it seem as if this were in fact the case. Time-sharing technology, theorized in the late 1950s and deployed by the early 1960s, allowed the tremendous cost of a computer to be shared by dividing the computer's time into infinitesimal increments. By spending a fraction of a second on one user's program, switching rapidly to other users's programs, then immediately moving back to the first program, it appeared as if the computer were responding

instantly to each user's commands. The position of each person's spaceship showed up on-screen in a split second, rather than several hours later; each user could think of the $250,000 computer in the back room as his or her own.

Contemporary scholars agree that the user was a modern invention; time-sharing systems, and the applications written for them, such as Spacewar, would invent the personal user.[5] But what is often lacking in these accounts is a description of the way the user's subject position is created not just by software, as media theorists would assert, but by the economic system that undergirds whatever relation any of us have with technology. If we take a closer look at the second part of Ceruzzi's sentence—the space warriors played "with no concern for how many ticks of the processor one was using"—we can sense the complexity of this economic model. Even as time-sharing allowed a new imagination of computing as a utility that could be billed down to each tick of a computer's clock, the game player would be encouraged to forget about that bill. Why bill and then erase the bill? It is an interesting paradox that is fundamental to today's digital culture, where downloads, storage space, e-mail, video, and so forth are free by default. Computer power is now so plentiful that we've stopped counting, and the minimal cost of computation seems to enable a sort of personal freedom. Computers are cheap enough, in other words, that we use them for blogging, playing games, personal expression, and so on—not just for work.

The paradox cannot be explained by software alone. We need another explanation, and this chapter revisits the history of time-sharing not to retell a well-worn story of inventors, technologies, and dates, but to show how time-sharing seemed to restructure the very boundaries between work and leisure, public and private. I analyze both the rhetoric invoked by computer scientists to describe their own work, as well as political, legal, and nonspecialist documents that set out a broader cultural imagination of what time-sharing could do. As I argue, time-sharing was part of a larger and more fundamental economic shift away from waged labor and toward what Maurizio Lazzarato terms the economy of "immaterial labor"—an economy of flexible labor that encompasses even seemingly personal or unpaid tasks, such as writing a review for a favorite product on Amazon.com.

By focusing on the time-shared user as an economic subject, we can understand many of the attitudes that structure present-day digital culture. For the irony is that though the word "time-sharing" went out of fashion with the advent of mini- and personal computers in the 1980s, the very same ideas have morphed into what seems to be the most modern of computing concepts: cloud computing. In cloud computing, time on expensive servers (whether storage space, computational power, software applications, and so on) can be rented as a service or utility, rather than paid for up front. Even the software housed on the Stanford PDP-10 has profoundly shaped the cloud: the mechanisms that once gave Spacewar players the illusion of having their own computer now gives a cloud's client the illusion of having a "virtual machine"—even when there are actually hundreds if not thousands of virtual machines all running on the same server.

After a roughly two-decade hiatus in which computing costs were generally bundled into the price of a new computer or a new software package, charging by computer time is back in vogue.[6] Cloud computing providers are again billing in computer-hours, and cloud software such as Office, Adobe Creative Suite, and so on is again rented in monthly or yearly subscriptions——a process pioneered by long-forgotten time-sharing businesses such as Allen-Babcock and Tymshare in the 1960s. In the first time-sharing systems, the $1.6 million cost of an IBM 7094 system in 1963 could be fractionalized to the tune of a mere $450/hour. As of 2014, Amazon was charging 45 cents per computer-hour on its Elastic Computer Cloud, or EC2: a thousandfold reduction from 1963 in nominal terms, and yet a service that is far more profitable for Amazon than books, DVDs, or groceries.[7] When futurists predicted that time-shared computers could become a public utility for millions, they were very nearly right; they were just several decades too early.

To understand the history of time-sharing, this chapter's first two sections follow two closely related tracks on the themes of intimacy and privacy. I begin by asking: how did computing come to feel personal? As I suggest, a sense of intimacy with the computer resulted from a user's flirtation with a series of transgressions: what was initially a voyeuristic relationship with an off-limits computer behind a glass wall (metaphorically, a "stolen look") set the stage for a modern operating system that accounts for each tick of "stolen time." Again, this shift was not simply the result of technological innovation; time-sharing was symptomatic of a postwar economic shift toward multitasking and freelancing. It is this larger context that explains why cloud computing deliberately confuses "free" time with liberal freedoms, and why it produces a quasi-illicit economy that encourages users to take (even steal) things for free. By positioning users as intimate partners of the computer, time-sharing yoked users to a political economy that made users

synonymous with their usage, and allowed them (or their advertising sponsors) to be tracked, rented, or billed down to each tick of the clock.

To make the intimacy of the user work, a user must be made to feel individual and private—even as millions of users share the same hard drives, computers, and data pipes underneath. A vast and unseen layer inside today's cloud, known as virtualization software, ensures that the data jumbled within the cloud's data centers and networks appear as individual streams of data (and each person's slice of a shared server appears as his or her own "virtual machine"). But individuation is also an ideology that plays a role far beyond the cloud's internal mechanism. For example, companies employ complex algorithms to filter content according to browsing history; when two people Google the same word, the algorithm will return different results to each, results that are meant to correlate to demographic information about each user.[8] As a result of this individuation, each user can be billed for actual computer usage—or, more commonly, each user's computer usage (now in the form of web history, clickstream data, etc.) can be turned into a commodity by advertisers. This ideology, to paraphrase artist Richard Serra, "delivers people";[9] the cloud has replaced television as the premier mechanism for sorting the public into private users.

How time-sharing systems "delivered" individual users and made them private is the subject of this chapter's second section. As part of this story, I explore how computer scientists working behind the scenes transformed the risks of many users sharing the same computers—whether computer viruses that afflicted overly "social" users, or garbage and unclean objects left behind in computer memory—from a set of catastrophic failures to an ongoing problem to be managed (and even, in today's "sharing economy," embraced). This mechanism is, however, a double-edged sword, for that mechanism also allows the cloud to exert a soft form of control over today's users. As I argue, the best way to understand how this power is deployed is not through surveillance cameras, web trackers, or cryptographic algorithms. Rather, it is an ancient technology, the sewer system: centuries before computers were invented, sewers kept each household's private business private even as it extended the armature of the state into individual homes. Using the case study of data leakage and leaking in general, the final section examines a possible way to reimagine this topography of control.

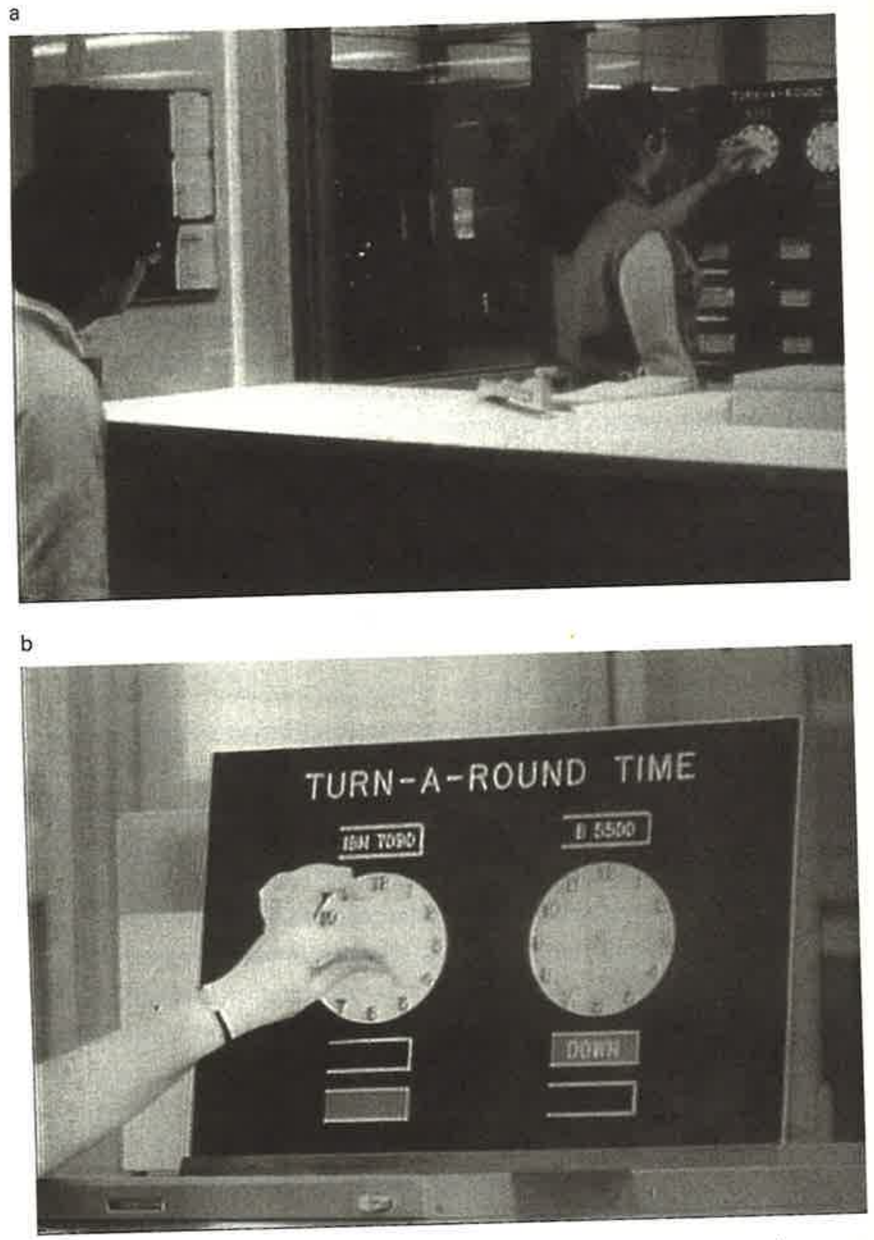

Figure 2.2a, b
Stills from Art Eisenson and Gary Feldman, *Ellis D. Kropotechev and Zeus, A Marvelous Time-Sharing System* (1967), 16mm film. Courtesy of the Computer History Museum; used with permission of Gary Feldman.

In a slapstick film produced by two Stanford graduate students in 1967, we are introduced to a mustachioed computer scientist Ellis D. Kropotechev, "a man with a problem, a girl, and a deadline."[10] Said girl, a painter, awaits him at a picnic, but because Kropotechev is stuck waiting for his punch cards to be processed, he is left in various stages of limbo: pounding the side of a jammed IBM punch machine, smoking in the cafeteria as he waits, and running down a never-ending hallway. After intertitles reading "Later" and "Still Later" and "Later Still" break up shots of Kropotechev's growing anxiety, we finally see the computer output returned to him. The printout shows the seconds of processing time billed, then an error message: the program has crashed. As we see a shot of the girl about to leave, Kropotechev races to deliver the next set of punchcards. But he slips and the punchcards fly out of his hand, Charlie Chaplin–style, fluttering uselessly across the screen. It is only through the intervention of a quasi-divine time-sharing computer named "Zeus" that our hero is saved—and his girl won.

The film is a humorous and brilliant introduction to the difference between batch processing and time-sharing. Filmed to the soundtrack of the Rolling Stones' "Cool, Calm, Collected" ("Well she's very wealthy . . ."), Kropotechev's object of desire, the painter, seems out of reach. What stands in Kropotechev's way is another woman: a computer operator, who collects the punchcards and sets a clock marked "Turnaround Time—IBM 7090" (figure 2.2a, b). This second woman metonymically recalls the fact that the word "computer" had originally designated the female operator of a machine, rather than the machine itself. As Jennifer Light explains, a computer was typically a low-paid laborer who initially served as a human calculator for wartime problems, such as ballistics, and soon took on the physically demanding work of programming circuits and fixing the machines herself.[11]

The manifest message of the film is an economic story: eliminating the middleman, speeding up the programming operation, and replacing the drudgery of low-paid labor with better technology so that pleasurable, higher-status activities—courting a "very wealthy" girl, painting—can occur. But there is a more complicated subtext, for if the goal of time-sharing is to free up time to get the girl, the real question is: which girl, the computer (operator) or the painter? The film pairs the two women so symmetrically—one the timekeeper, one the expiring deadline—that the economic benefits of time-sharing are embodied in sexual terms. One computer delays and prevents climax; the other (named, by delightful coincidence, after a Greek god known for his many dalliances) enables it. The virility bestowed by the time-sharing system allows Kropotechev to bypass the drudgery of batch processing, fend off a romantic rival, and reclaim his girl. (This nominally heterosexual narrative becomes a bit queer in the substitution of machine for flesh, as when the Zeus computer demonstrates that it, too, has a drawing and painting program, just like Kropotechev's object of desire.[12]) These tangled sexual undercurrents within real-time programming aptly illustrate why Fernando Corbató, deputy director of MIT's Compatible Time-Sharing System (CTSS), described interacting with his time-sharing system as "programming intimacy."[13]

Corbató was not the only computer scientist to think about time-sharing in these terms. J. C. R. Licklider opened his seminal article "Man-Computer Symbiosis" (1960) with a parable about the "very close coupling" of an insect that lives in the ovary of a fig tree and in turn pollinates it; man and computer, he writes, are two creatures "living together in intimate association, or even close union."[14] The word "intimacy" has now been superseded in contemporary parlance by terms such as "interactivity," but it is a useful one, as it allows us to recover a largely sublimated sense of desire within the concept of the user. This relationship between human and machine predates the computer: the typewriter, Friedrich Kittler similarly observed,[15] once designated a female secretary and now refers to a machine. And it occasionally becomes explicit, as in the crude joke that "programming is like having sex: one mistake and you have to support it for the rest of your life." But this sensibility has largely been lost or disavowed.

I use the lens of intimacy to revisit time-sharing, an approach that allows me to examine how the always unstable desire for the computer—whether (human) operator or machine—creates the subject position we now call the user. To explain the subjectivity of early computation before time-sharing, one might find an analogy to cinema, which constructed a spectacle for viewers to look at—but always at a remove. If the quasi-illicit pleasure of cinema is peering in on a scene, as if through a keyhole, the computer user initially occupied a similar position. For their programmers, the batch-processed computer was visible but physically out of reach: at Stanford, as *Ellis D. Kropotechev and Zeus, A Marvelous Time-Sharing System* illustrates (figure 2.2a), the IBM 7090 was inside a glass room guarded by computer operators. Sometimes called the "glass house," this machine room (cold, noisy, and a generally forbidden space) both walled off the computer even as it occasionally

revealed it; though programmers were required to submit program reque a to the staff, they could often steal looks at the computer's internal registers through the glass.

The idea of looking was also a long-standing metaphor for the act of programming—which is to say debugging—itself. As computer historian Martin Campbell-Kelly writes, before the time of the IBM 7090: "There were, of course, no software debugging aids whatever on the EDSAC at this time [1949], so the program had to be debugged on a naked machine, by 'single-stepping' through the program and observing the contents of the memory and registers on the monitor tubes. This process was known by the rather charming name of 'peeping.'"[16]

Put differently, debugging in the era of batch processing was an explicitly voyeuristic act. The act of "peeping" into a computer in the middle of its routine constructed a scopophilic relationship between user and machine. Even today, debuggers occasionally use the word "peeping": one guidebook to Visual Basic .NET announces that the "Watch window also allows you to be a peeping Tom."[17] (The window of modern graphical operating systems is therefore conceptualized not just as a portal into a virtual world elsewhere, but also a window into the inner secrets of the machine.[18])

As a sort of voyeur, the batch-processed user's exact identity was not particularly important: each user was interchangeable with another from the standpoint of the computer. Consequently, there were no individual user names, and the user was merely a number recorded for billing purposes. The gap between man and machine thus remained a perverse distance, one whose occasional crossing recreated the enjoyment of a Peeping Tom.

This gap, however, began to be bridged in a series of technologies that culminated in time-sharing. The first of these technologies came in the 1950s, when crashing programs triggered a small routine known as the "post-mortem dump." After the death of the user's program, this trigger printed out the previously interior contents of a computer's memory so that the user could study why the program produced an error. As an automatic if indiscriminate way of inspecting the computer's interior, the dump was a literal manifestation of both waste and a user's time wasted. Waiting for the operator to load the next program, the user waited several hours before trying again, in a cycle that represents a failure of consummation, or at least, of elimination: "Several attempts must be made before all errors are eliminated. Since much machine time can be lost in this way a major preoccupation of god known for his many dalliances) enables it. The virility bestowed by the time-sharing system allows Kropotechev to bypass the drudgery of batch processing, fend off a romantic rival, and reclaim his girl. (This nominally heterosexual narrative becomes a bit queer in the substitution of machine for flesh, as when the Zeus computer demonstrates that it, too, has a drawing and painting program, just like Kropotechev's object of desire.[12]) These tangled sexual undercurrents within real-time programming aptly illustrate why Fernando Corbató, deputy director of MIT's Compatible Time-Sharing System (CTSS), described interacting with his time-sharing system as "programming intimacy."[13]

Corbató was not the only computer scientist to think about time-sharing in these terms. J. C. R. Licklider opened his seminal article "Man-Computer Symbiosis" (1960) with a parable about the "very close coupling" of an insect that lives in the ovary of a fig tree and in turn pollinates it; man and computer, he writes, are two creatures "living together in intimate association, or even close union."[14] The word "intimacy" has now been superseded in contemporary parlance by terms such as "interactivity," but it is a useful one, as it allows us to recover a largely sublimated sense of desire within the concept of the user. This relationship between human and machine predates the computer: the typewriter, Friedrich Kittler similarly observed,[15] once designated a female secretary and now refers to a machine. And it occasionally becomes explicit, as in the crude joke that "programming is like having sex: one mistake and you have to support it for the rest of your life." But this sensibility has largely been lost or disavowed.

I use the lens of intimacy to revisit time-sharing, an approach that allows me to examine how the always unstable desire for the computer—whether (human) operator or machine—creates the subject position we now call the user. To explain the subjectivity of early computation before time-sharing, one might find an analogy to cinema, which constructed a spectacle for viewers to look at—but always at a remove. If the quasi-illicit pleasure of cinema is peering in on a scene, as if through a keyhole, the computer user initially occupied a similar position. For their programmers, the batch-processed computer was visible but physically out of reach: at Stanford, as *Ellis D. Kropotechev and Zeus, A Marvelous Time-Sharing System* illustrates (figure 2.2a), the IBM 7090 was inside a glass room guarded by computer operators. Sometimes called the "glass house," this machine room (cold, noisy, and a generally forbidden space) both walled off the computer even as it occasionally

revealed it; though programmers were required to submit program requests to the staff, they could often steal looks at the computer's internal registers through the glass.

The idea of looking was also a long-standing metaphor for the act of programming—which is to say debugging—itself. As computer historian Martin Campbell-Kelly writes, before the time of the IBM 7090: "There were, of course, no software debugging aids whatever on the EDSAC at this time [1949], so the program had to be debugged on a naked machine, by 'single-stepping' through the program and observing the contents of the memory and registers on the monitor tubes. This process was known by the rather charming name of 'peeping.'"[16]

Put differently, debugging in the era of batch processing was an explicitly voyeuristic act. The act of "peeping" into a computer in the middle of its routine constructed a scopophilic relationship between user and machine. Even today, debuggers occasionally use the word "peeping": one guidebook to Visual Basic .NET announces that the "Watch window also allows you to be a peeping Tom."[17] (The window of modern graphical operating systems is therefore conceptualized not just as a portal into a virtual world elsewhere, but also a window into the inner secrets of the machine.[18])

As a sort of voyeur, the batch-processed user's exact identity was not particularly important: each user was interchangeable with another from the standpoint of the computer. Consequently, there were no individual user names, and the user was merely a number recorded for billing purposes. The gap between man and machine thus remained a perverse distance, one whose occasional crossing recreated the enjoyment of a Peeping Tom.

This gap, however, began to be bridged in a series of technologies that culminated in time-sharing. The first of these technologies came in the 1950s, when crashing programs triggered a small routine known as the "post-mortem dump." After the death of the user's program, this trigger printed out the previously interior contents of a computer's memory so that the user could study why the program produced an error. As an automatic if indiscriminate way of inspecting the computer's interior, the dump was a literal manifestation of both waste and a user's time wasted. Waiting for the operator to load the next program, the user waited several hours before trying again, in a cycle that represents a failure of consummation, or at least, of elimination: "Several attempts must be made before all errors are eliminated. Since much machine time can be lost in this way a major preoccupation of the *EDSAC* group at the present time is the development of techniques for avoiding errors, detecting them before the tape is put on the machine."[19]

If "peeping" positioned the user as a voyeur who secretly gazed on the computer's inner operations with a stolen look, improved debugging technologies reshaped this relationship through a newly transgressive pleasure: that of "stealing" computer time. Early experiments in the late 1950s by John McCarthy, Steve Russell, and Herb Teager explored the potential of what they called "time-stealing."[20] This process allowed an important professor's job to temporarily interrupt an existing computer program. After the success of this demonstration, McCarthy and others convinced computer manufacturers at IBM and Digital Equipment Corporation to modify their memory systems to support time-sharing, and the first systems began to arrive at MIT in the early 1960s.[21]

As time-stealing technologies matured into time-sharing, they seemed to miraculously recover time that was hitherto wasted, allowing each user to steal computer cycles out of thin air. In 1966, for example, Douglas Parkhill devoted a chapter of *The Challenge of the Computer Utility* to estimating the unused capacity of government computers. Sixty percent of computer time is wasted, he speculates, resulting in a potentially gargantuan $550 million of savings if time-sharing were deployed.[22] Parkhill probably did not count on home-brewed programs filling these empty cycles. Yet time-sharing led to an explosive growth in these informal and unanticipated uses. Because cycles could now be run on the side, this "two-timing" aspect of time-sharing opened a variety of ways for users to furtively acquire time. One could bill one's program to the operator account, forge the books, or zero out one's billing account.[23]

Surprisingly, managers at MIT did not clamp down entirely on users' ability to use time-sharing systems for themselves. Instead, they came up with a way of regulating the system that is now so naturalized as to seem obvious, but was completely novel at the time: naming each user. As we saw earlier, users were simply problem numbers or terminals in a batch-processing system, but time-sharing put forth a mental model of the user who had both a username and a password. The goal, Corbató recounted, "was to make it personal so that we were dealing with individuals . . . we also needed accounting. It wasn't so much for charging but [to] keep track of usage at least."[24]

To make computing "personal," to be "known as a person when you logged in," the user had to be radically refashioned. Crucially, the user was not yet

personal in the sense of personal computing; rather, this process made a user's personhood synonymous with his or her *usage*. It mapped the identity of the user onto his or her time spent using the computer. Many hackers took umbrage at what they saw as a managerial intrusion: "People had to get accounts and had to pay attention to security. It was a benign bureaucracy, but nevertheless a bureaucracy."[25] Making this need to register a user's identity explicit, a 1968 ARPA study tested twenty-one programmers on a mix of time-sharing and batch-processing systems to gauge whether time-sharing would help with programming error. The scientists carefully logged and tracked each user's person-hours worked "by close personal observation," and compared the figure with the number of computer-hours used; "discrepancies . . . were resolved by tactful interviewing."[26] The study thus indicates an early attempt to reconcile the "real" number of hours worked with the hours used by the computer.

By making the user equivalent to his or her usage, time-sharing yoked the user's labor to the labor of the computer itself. In doing so, human–computer interaction initially functioned as a management technique, as a way to fashion an efficient worker capable of flexibly managing time. (It is no coincidence that multitasking, a concept that originally came out of time-sharing, now refers to this kind of flexible work.) This new kind of worker resulted from a broader economic shift away from factory-based work and toward immaterial labor, in which "workers are expected to become 'active subjects' in the coordination of the various functions of production, instead of being subjected to it as simple command" (Lazzarato).[27] This type of economy most fully manifests itself through a cybernetic model of control, in which user and computer jointly make decisions in real time. Thus, scientist Licklider explains, the user would be asked to "fill in the gaps . . . when the computer has no mode or routine," and similarly, the computer would perform "clerical operations that fill the intervals between decisions."[28]

Despite naming it "real time," the mode of time that Licklider describes is neither "real" nor unmediated; real time actually functions as an ideology of economic productivity. By splitting a problem into thousands of increments, and then stitching these intervals of computer and worker time alike back into a seeming whole, the computer disavows unproductive moments with "no mode or routine," and turns our attention away from these gaps, stutters, and freezes and toward more productive modes of work.[29] Yet to do so is to subtly refashion the subject brought within the domain of real time. To understand this, consider a parallel example of a film spectator, which is also subjected to numerous gaps within and surrounding the film: for instance, a cut from one scene to the next, the inactive and unseen moments between two moments of action, or even, in the case of analog film, the black leader in between film frames.[30] Even as we are repeatedly subjected to these gaps, a set of methods, termed "suture" by film theorists, creates a subject position that offers the illusion of unity.

"The operation of suture," Kaja Silverman explains, "is successful at the moment that the viewing subject says, 'Yes, that's me,' or 'That's what I see.'"[31] Similarly, with a program sent into limbo and then reanimated every fraction of a second, a computer's operating system also employs techniques to ensure a user does not notice the gaps. Thus, when a computer user can stand in as an "I"—as a unified subject—the operation of real time succeeds. And yet, in actuality, a computer's "real time" is a process of reassembling thousands of seconds of time "stolen" from other programs. Because time-sharing equates a computer user with the amount of work done by the computer, the user is actually an assemblage of economic value—with the time spent using a computer a commodity to be tracked. Though similar to marketing methods that monetize a viewer's television watching time, there is one seeming difference: the user is expected to interact, to actively "use" the computer.[32]

Like a sleight-of-hand trick, then, time-sharing performed two things simultaneously: it created a sense of personal intimacy with the computer, even as it masked the economic mechanism that supported it. To recall Spacewar, it made users feel as if they were playing "for fun, interactively, with no concern for how many ticks of the processor one was using." This trick was not necessarily a bad thing; time-sharing was often experienced as a permissive and even freeing system, for it asked users to think of the computer's time as their own time. Still, by connecting each user to a central computer, and by implication, the work of the computer, the link that bound the two in unison would increasingly function as a subtle tether, a motif that will recur throughout the remainder of this chapter.

From a contemporary perspective, the intimacy experienced by a computer user may seem to be the first step toward liberal subjectivity—with computers as "cyborg partners, second selves, a new subjective space that included the machine" (Paul Edwards).[33] It may also appear that this intimacy was an inevitable result of technologies for human-computer interaction.

But this is an oversimplification; time-sharing was initially developed for debugging, not for interactivity.[34] Moreover, time-sharing was not universally acclaimed; some programmers resented the need to be present at the same time as their programs. (Time spent waiting for a batch-processed computer to return results undoubtedly provided a welcome break from the routine of work.) And most important, we shouldn't forget that time-sharing was first and foremost a new way of conceiving and accounting for one's own time in relationship to productivity. Any sense of the "personal" relationship that began to develop between people and computers was a symptom, a secondary and holographic effect.

As part of this shift, time-sharing normalized and even promoted the transgressive potential of time stealing. Because typewriters or TV-like displays now connected users to the computer, there was no need to steal looks at the computer itself; accordingly, the real computer moved out of the glass-walled rooms and out of sight. Users could now "steal" computing power when needed—both taking processor time while doing something else, as if stealing away from work to run a personal errand, and also taking advantage of processor time as if it were free. In this, time-sharing anticipated the way that the contemporary cloud encourages its users to take things free of charge. By making each online resource freely available—computer storage, processing time, content, even software—the cloud encourages the pleasurable and quasi-illicit feeling that we are getting away with something: that we, too, have stolen time.

To unpack this idea, let us return, once again, to Spacewar. Recall that the Stanford gamers played "for fun, interactively, with no concern for how many ticks of the processor one was using." Their knowing disavowal of the cost of computing time—a PDP-10 may still have cost half a million dollars in 1972, when Brand wrote his article—evokes the contradictory resonances of the word "free." On first glance, free may suggest a space located outside the marketplace—as merely "for fun," with no commercial value.[35] But free also takes on the sense of the phrase "free time": time off from work, perhaps, but only within a labor market where play can pay off and work may seem like play. Thinking of his employees playing Spacewar, Les Earnest, executive director of Stanford's AI Lab, commented: "Sometimes it's hard to tell the difference between recreation and work, happily. We try to judge people not on how much time they waste but on what they accomplish over fairly long periods of time."[36]

Earnest's words remind us that, in today's Silicon Valley, recreation and personal interests have become subcategories of work, with even recreational activities and time often seen as ways of furthering a company's productivity.[37] By agreeing to work in real time, the user enters a flexible economic framework in which she is free to choose how to spend her time, as long as time is understood as something to be spent. While the batch-processed user was primarily an accounting method that counted the number of computer hours to be billed for each problem, time-sharing asked users to *account for themselves*.

As a consequence, what we think of the "user" confuses personal intimacy for economic intimacy. This confusion may explain why so much of digital culture is powered by user labor and user-generated content. Laboring in a time-shared economy—everything from tagging a photo on Facebook to reviewing businesses on Yelp to answering product questions on Amazon—is performed for the love of the task, for personal reasons and during "free time," even as this labor generates value (if not profit) for the company that administers it. Theorist Tiziana Terranova enumerates this sort of work: "Simultaneously voluntarily given and unwaged, enjoyed and exploited, free labor on the Net includes the activity of building web sites, modifying software packages, reading and participating in mailing lists and building virtual spaces."[38] To be sure, free laborers are by no means dupes; indeed, the gentle tether of a time-sharing economy most closely resembles the freedom of a freelancer—the ability, in other words, to decide for themselves which projects to take on, which activities will pay off, and which projects are personal ones.[39] Once again, this flexibility can often be genuinely experienced as liberatory. But this flexibility comes, as they say, at a price.

The bargain is thus: to be a user, you must continually act (and act within this framework). As Lazzarato comments, postindustrial capitalism takes as its slogan "Become an active subject": "one *has to* express oneself, one *has to* speak, communicate, cooperate, and so forth."[40] Conversely, the one user of no value to online companies is the user who fails to "use," who registers with a website and then never returns. This failed user, the user that doesn't participate or produce content, represents the queer stoppage of technological (re)productivity.[41] Consequently, much of the free storage or free offerings in today's digital culture is structured to entice and reward (if not compel) participation. As time-sharing's successor, the cloud is the fullest manifestation of phenomena described as "freeware capitalism": "[Free stuff] makes

you just consume more time on the net. After all, the goal . . . is to have users spend as much time on the net as possible, regardless of what they are doing. The objective is to have you consume bandwidth."[42]

Regardless of how much bandwidth costs, and how much actual money is spent, the underlying logic of freeware capitalism is consumption—of time. While it is far too easy to critique today's monetization of the Internet user, these arguments imagine an originary Internet gift economy that is purely fictional. The reality is that ever since time-sharing systems bestowed names upon users, those users have been interpellated as units of economic value.

This brings me to a closely related point. Complaints about monetizing the Internet user inevitably raise the specter of a user's privacy being violated: buying and selling clickstream data, tracking, and so forth. Privacy debates typically invoke the rhetoric of intimacy to explain this sense of violation. Pictured as an algorithm "peeping" into a computer user's bedroom, surveillance in the cloud seems to involve the interplay of voyeurism, watching, and stolen looks. Meanwhile, the metaphors of sexual relationality are displaced on the realm of technological reproducibility: for example, the promiscuous proliferation of copying and file sharing.[43] When privacy advocates invoke a so-called "right to privacy" online, they reenact and reanimate the libidinous circuits of desire that run through US Supreme Court decisions on "whether to bear or beget a child . . . personal decisions related to marriage, procreation, contraception, family relationships, child rearing, and education."[44] The topics of the landmark privacy cases cited as a precedent for digital privacy law therefore range from contraception (*Griswold v. Connecticut*, 1965) to viewing pornography (*Stanley v. Georgia*, 1969) to abortion (*Roe v. Wade*, 1972) to homosexuality (*Lawrence v. Texas*, 2003).

Fixated on these problems of the stolen look and personal intimacy, these debates misread the digital user as a liberal subject. But in the vast majority of the cases, users negotiate their privacy "freely" (which is to say as freelancers): working within the confines of private contracts, such as software licensing agreements. These matters therefore fall within the realm of tort law, rather than constitutional law. Asked to manage ourselves as users, we are also asked to manage our own privacy online by negotiating with private companies: by taking on liability for copyright infringement on reposted images, by managing whom our posts can be shared with, and even by opting in to restrictions on search results for children, in what Raiford Guins terms a culture of self-imposed filtering.[45] The subtext here is that in return, we generally expect some say in how our data should be monetized: when to run advertisements, for instance, on our "personal" blog. When this contract seems to fall apart—when a social media company sells an image that we have produced, without our permission—we feel it as a violation of personal intimacy, even when the damage is, in actuality, an economic one.

The irony is that many digital privacy debates are attempts to resurrect a Victorian ideal that imagines a separation between the private and the public spheres.[46] But the idea of privacy on a time-shared system would have been difficult to comprehend in the 1960s, when mechanisms to produce a sense of user privacy were not yet fully developed. After all, despite a user's intimacy with "his" or "her" computer, that same computer was shared with tens to hundreds of other users (and now, in the cloud, thousands). As technology journalist Steven Levy wrote, "The very idea that you could not control the entire machine was disturbing . . . you would just know that it wasn't all yours. It would be like trying to make love to your wife, knowing she was simultaneously making love to six other people!"[47] The separation with public and private did not yet exist; the boundaries of intimacies first needed to be built. Analogous to the technologies of government and economic management that produced a conception of the self in the period we now call modernity, the user itself had to become a "modern user," one that seemed private and individual even as it was positioned wholly within a time-shared economy.

In what follows, I circle back to time-sharing's origins in the 1960s to offer a second story, of how the "benign bureaucracy" of system managers made users private. This narrative will closely track the one we have just considered on development of a time-shared economy. These two methods for governing users—first, by promoting computer usage as a vehicle for economic intimacy, and second, by preventing too much intimacy between users—both do their work behind the scenes. They act, like a carrot and stick, as a hidden layer of control within the mechanisms of cloud computing.

"The Victorians Built Magnificent Drains": Waste, Privacy, and the Cloud

Almost as soon as they were deployed, time-sharing systems began to reshape the social compact between their users. There were other people around at the same time, and in response, users quickly began to test the boundaries

of social acceptability. Scientist Alan Kay describes the early days of MIT's Project MAC (Multiple Access Computer), circa 1963: "One of the guys wrote a program called 'The Unknown Glitch,' which at random intervals would wake up, print out I AM THE UNKNOWN GLITCH. CATCH ME IF YOU CAN, and then it would relocate itself somewhere else in core memory, set a clock interrupt, and go back to sleep. There was no way to find it."[48]

The best symbol for this new era was therefore less Spacewar than Core Wars—battling a glitch, bug, or infection from someone or something within the computer. Still, in university labs, these actions did not yet rise to the level of a violation. In a single phrase, Baran perfectly captured this inchoate sense of digital privacy as fooling around under the covers: "hanky-panky."[49]

All this would change in the mid-1960s, when the secret of time-sharing left the labs and burst onto the public's attention. Time-sharing systems had expanded so rapidly that leading computer scientists envisioned it would become a public utility, like the telephone system.[50] (This vision remains almost unchanged today, except that cloud computing is more accurately described as a series of *private* utilities.) Martin Greenberger had presciently predicted the "computer utility" in a 1964 essay for *The Atlantic*, and a 1965 issue of *Time* was among the first to tell businessmen that "'time sharing' is part of a growing trend to market the computer's abilities much as a utility sells light or gas."[51] Early 1966 saw the publication of a profile of MIT's Project MAC in *Scientific American* and Parkhill's full-length book *The Challenge of the Computer Utility*.[52] In the same year, the chairman of the US Service Commission excitedly informed the public that his agency was planning a new National Data Center that would collect and centralize data from twenty-two government departments, such as the Census Bureau, the Internal Revenue Service, and Social Security, in the name of efficiency and automating government.[53]

This last application seemed to merely extend moves by local government into time-sharing. Municipal areas such as Alexandria, Virginia, Tulsa, Oklahoma, Little Rock, Arkansas, Fort Worth, Texas, Denver, Colorado, and Detroit, Michigan, had already established "Metropolitan Data Centers" and other urban data banks to share demographic information, while the groupings only expanded: "a central time-shared computer system for twenty San Gabriel Valley (California) cities with sharing of data of common interest to several cities."[54] But something about the federal government taking charge—and the mingling of criminal, tax, census, and Social Security records—struck a nerve. Only a few weeks after the article announcing the

National Data Center hit the newsstands, an alarmed congressional subcommittee convened hearings on "The Computer and Invasion of Privacy" with a rapidity that caught even computer scientists by surprise.[55]

Held over three days from July 26–28, 1966, the hearings featured law professors, sociologists, computer scientists, and New York state officials who had built a smaller version of this proposed data center. The atmosphere of consternation had a heavily libertarian bent: George Orwell's *1984* and references to police states featured repeatedly, and the record contains an excerpt from John Stuart Mills's "On Liberty." One computer scientist claimed the development of data centers would be as "potentially dangerous and powerful as a nuclear explosive device";[56] another expert warned that at the rate of technology's development, there would not be much left for taxpayers to celebrate by the time the United States reached its bicentennial. Despite this rhetorical posturing, the hearing was remarkably ahead of its time. It referenced strategies that closely resemble today's headlines about data privacy: "information relational retrieval" techniques that infer relationships between people using metadata; automated data mining that would reduce the "cost per unit of dirt mining by unautomated human garbage collectors," and thus smear reputations for pennies on the dollar; even speculation of a data-personhood determined by a computer trawling through records as diverse as "book clubs [and] magazine subscriptions."[57]

This hearing indexed a moment of transition from user as programmer in a computer lab to user as part of a population. While these data banks and "computer utilities" anticipated the data centers and the always-on utility of today's cloud, they were still inchoate in form. What the future might look like would still be very much up for grabs. Advocates therefore sought explanatory parallels for its risks in previous public infrastructures. Testimony from the congressional hearing compared it to other "natural monopolies," such as the nineteenth-century infrastructures of the railroad and the telegraph; earlier, Greenberger had also drawn a parallel between the new computer utility and the electrification of cities by describing the gradual replacement of gas lamps with Edison electric streetlights in the early 1900s.[58] Two analogies emerge from this hearing, however, that are of particular interest to our story: the telephone line and the plumbing system. These analogies offer a way to understand veiled cultural attitudes toward computing that may not emerge from a more direct look, and I consider them in sequence below.

Of the two analogies, the telephone line is perhaps the most obvious precedent: wire taps on a telephone line (even when there is no physical line any more) continue to be the legal standard by which US courts consider digital eavesdropping cases. But one twist emerges in the context of time-sharing. One legal scholar compared the privacy concerns of sharing a computer with sharing conversations on a party line.[59] The unwritten subtext to his example is a generally forgotten cultural shift in the way that telephone lines were imagined. Eavesdropping on party lines did not used to be considered an invasion of privacy, historian Ronald Kline tells us; indeed, eavesdropping was a widely accepted practice in rural communities that often formed that community's "social network." It was for economic reasons—eavesdropping drained the batteries in their equipment—that telephone companies developed public awareness campaigns to teach their users not to eavesdrop. These campaigns described the social strife that resulted from eavesdropping and shamed the men who listened nevertheless by likening them to thieves—or gossiping women.[60] Taking the form of poems, films, and comic strips, these lessons in party line etiquette served to recast social behavior as antisocial behavior, as in *Bobby Gets Hep*, which describes a teenage boy, Bobby, whose penchant for tying up the party line leads to a barely avoided disaster (figure 2.3).

Now we consider eavesdropping on a telephone an antisocial act, a violation of privacy, but what has actually changed is the conception of the *user*. Similarly, what is considered digital privacy is formed less by the technology itself than the social codes or norms around the user that produce its individuality. This section explores the progressive development of these codes in time-sharing systems that made the user seem individual—and, reciprocally, made computing an individual matter. There is no single defining moment or cultural artifact that can encapsulate this development; IBM ran public awareness campaigns in the 1980s that bore some resemblance to AT&T's campaign, but a variety of technologies and strategies have also shaped this discourse. The specific threats to computer privacy—wiretapping, computer viruses, and, now, the bulk leaking of data repositories, whether by state or nonstate actors—have varied over the last fifty years, and the strategies have changed in response. But by the end of this section, we shall see that these strategies, considered in aggregate, grow out of the intersection of political and economic discourses about the proper role of a state to its subjects.

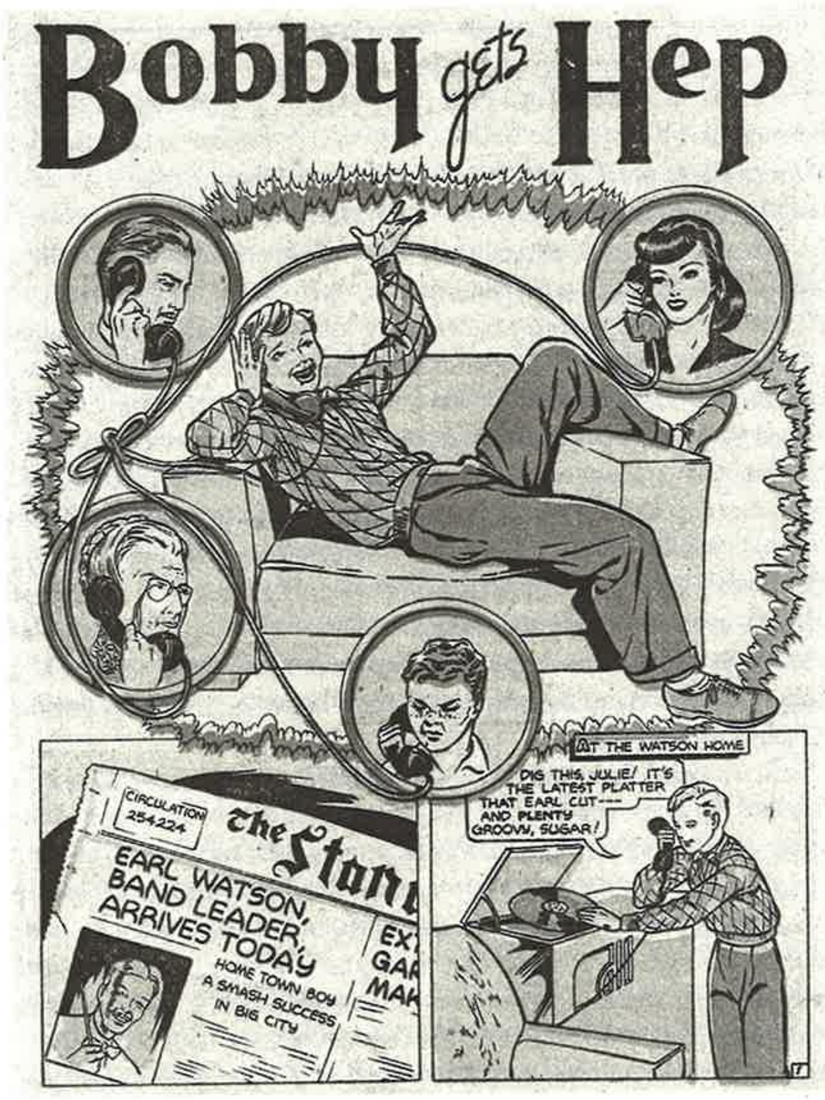
Bobby gets Hep
AT THE WATSON HOME
DIG THIS, JULIE! IT'S THE LATEST PLATTER THAT EARL CUT— AND PLENTY GROOVY, SUGAR!
CIRCULATION 254224
The Stan
EARL WATSON, BAND LEADER, ARRIVES TODAY
HOME TOWN BOY A SMASH SUCCESS IN BIG CITY
EX
GA
MAK
1

Figure 2.3
Bobby Gets Hep, Bell System comic for teaching party line etiquette, 1946. Scan courtesy Ethan Persoff, http://www.ep.tc.

In this light, it is worthwhile to note that strategies for imagining (and producing) a private user at the 1966 hearings resembled the governmental process of urban planning. Because of the number of municipalities considering data banks at the time, urban planners were an integral part of the hearing's audience, and, reciprocally, they invited computer scientists working on time-sharing to urban planning conferences.[61] Comments media archeologist Jussi Parikka: "[The] concern was how to fit dozens of people within this electronic 'space'—a problem that was analogous to the general problem of modernity in cities: how to deal with the issue raised by a huge number of people living in condensed urban spaces? . . . what is overgrowth, waste or a social problem"?[62]

It is in this context that we find our second analogy for understanding computer privacy. Testifying at the hearings, Baran inadvertently stumbles on a potent metaphor: the sewer. The safeguards required to protect privacy, he argues, may only come to be built during a moment of crisis: "We have, for example, been practicing [safeguards] in the design of sewerage systems and in electrical distribution systems for some time. But, historically, it usually has taken an epidemic to build a local sewerage disposal system."[63] Baran's description was quite prescient: waste—and specifically, the risk of contamination, uncleanliness, or infection—becomes a recurring leitmotif in discussions about privacy and time-shared computers. Legal scholar Arthur R. Miller singled out time-sharing in a section of "Personal Privacy in the Computer Age," writing about the damaging possibility of a "residuum of one customer's information accessible to the next user who is placed in control of the heart of the machine."[64] And citing Baran's example of automated dirt collection, a second legal scholar, Kenneth Karst, anticipated today's parlance on data leakage as he writes: "The risks of leakage in a shared-time system are obvious."[65]

Waste is particularly apt because it allows us to understand the effect of what Parikka terms "digital hygiene" in today's digital culture: the idea that a user is responsible for keeping her data from mixing with others, for avoiding infection with computer viruses, and so forth. Hygiene, of course, is not only produced by infrastructure, such as sewers, but is also a historically specific practice for the exercise of power. If we now think we have a direct and unmediated relationship to our own hygiene (making, for example, the privy inextricable from privacy), this is the distant legacy of a process of modernization. As Dominique Laporte tells us in his *History of Shit*, the individualization of hygiene may be traced to the modern state enjoining its subjects to keep (and bury) their own shit within the bounds of their homes.[66] Rather than dumping one's waste into the streets for all to see, the state interpellated the household as a private unit for managing one's own domestic life.

Baran's own reference to the sewer is fleeting; a year later, he will normalize it by comparing the computer utility to something that can "pipe computer power into homes," a metaphor that continues to be used to this day.[67] Yet it is through this lens of waste management that we can excavate a buried history of managerial control within the cloud. If, as Laporte puts it, "Surely, the State is the Sewer," we can discover how the state exercises power over its users—and how the individual user was produced—by revisiting computer history through the lens of waste management. How, then, should computer users be kept private; how should they be managed as a population? To answer this question, I offer two case studies of social "risks" that, at various moments of time-sharing, seemed to threaten the well-being of users as a whole: programming errors and computer viruses.

Case 1: Programming Error

Earlier, we saw that time-sharing systems developed as a response to programming errors and wasted time debugging. In a follow-up to "Man-Computer Symbiosis," J. C. R. Licklider and Robert Taylor offer a tongue-in-cheek vision of a future where there are so many programming bugs that "unemployment would disappear from the face of the earth forever" as everyone turns into a programmer: "the entire population of the world is caught up in an infinite crescendo of on-line interactive debugging."[68] As the number of users sharing each system multiplied, each program's errors could cascade onto those of other users, slowing down their programs and causing the system itself to crash. Errors, in short, could grow to the point where they would affect more than the initial user; they could form a kind of epidemic. And initially, the solutions were not subtle; the 1968 study that tested the effectiveness of programmers concludes: "validated techniques to detect and weed out these poor performers could result in vast savings in time, effort, and cost."[69]

These "poor performers" may well have been culled, but most operators found gentler ways of disciplining wayward programmers. Programmers were trained to follow what one scientist termed "defensive programming

techniques."[70] Additionally, an automatic routine—termed the system monitor, supervisor, or executive—kept track of the hardware resources used by each user, audited their program executions, and kept a user's program from erroneously writing over or otherwise interfering with other users' programs. Writes Licklider: "There are even arrangements to keep users from 'clobbering' anything but their own personal programs."[71] The predecessor to a modern-day operating system, the monitor was also the only program empowered to perform certain actions, such as writing to output.

The monitor's protections acted as a sort of privacy barrier between each program, and, consequently, between each user. But a draconian system monitor was the subject of protest by one programmer, who likened these behaviors to governmental overreach: "This is bureaucracy run rampant and a protest is in order . . . May I lodge a violent and heartfelt protest against this unwanted and unfair interference by the monitor with my rights as a programmer!"[72] His editorial was published in *Communications of the ACM* under the title "Ye Indiscreet Monitor," a clear evocation of the way that the monitor seemed to "peep" in on each program as it was running.

Inside the metaphorical city of electronic space, the monitor was a draconian system of governance—one more benevolent than culling individual users, perhaps, but nevertheless felt as a violation of privacy. Moreover, the "indiscreet monitor" was not always effective in reducing error rates and wasted resources, leading computer scientists to look for other techniques. In 1959, the year that John McCarthy proposed the time-sharing modifications discussed earlier, McCarthy also described the idea of "garbage collection," in which so-called dead objects (memory that is no longer in use or "live") would be automatically culled.[73] Garbage collection served as a safety measure that could reduce programming errors and memory leakage between different programs. It improved the system's overall performance, because it would not be slowed down and choked by too many dead objects. Programmers would, ideally, not even notice when the garbage collector came by (though, in practice, one felt a several-second delay as the entire system paused); ideally, one would program without noticing the garbage that one produced or the resources one used.

Working in the background, garbage collection was a more "discreet" method of managing users than the monitor—even if it attempted to subtly redirect and change their very approach to programming. It was a positive, behavioral measure that did not punish or restrict a specific user, but sought