정보 사유화의 울타리 걷어차기: 자유소프트웨어운동과 그 이상

조동원[1]

목차

0. "리룩스는 뭐하는거야?"

마이크로소프트가 소유하고 있는 독점 운영체제(OS) 소프트웨어인 윈도의 새 판올림, '윈도7'이 출시된 지 보름 정도가 지난 2009년 11월 4일, "윈도7, 호환성 불만 '봇물'"[2]이라는 보도가 있었다. "아래아한글, 와이브로 단말기, 인터넷 메신저 프로그램 등 주요 프로그램이 호환되지 않아 이용자 불편"이 많다는 것이다. 많은 댓글이 달렸는데 대체로 그 이전판인 '윈도XP'를 잘 쓰고 있는데 뭐하러 새 판을 깔았냐는 의견들과 함께 마이크로소프트에 대한 성토의 내용이 많았다. 그 중에 하나는 "리룩스는 뭐하는거야?"라는 제목이다:

> 회사에 누워서 과자 까먹고 있는거야? 이늠의 리눅스회사는 어떤 바보가 투자하길레 돈만 퍼붓고 있는거야. ms에서 몇년째 미친짓하고 있는데 아직도 올라서지 못하는거야.

이 글은 이 댓글에 대한 하나의 답글로서 그누/리눅스[3]가 뭐하고 있는지를 다룬다. 우선, 소프트웨어라는 정보 생산물이 정보 상품으로 되면서 마이크로소프트가 "몇[십]년째 미친짓"을 할 수 있게 된 배경에 대해 조금 이야기하고, 그누/리눅스와 같은 자유소프트웨어운동[4]이 만들어온 대안의 함의들을 짚어본다. 그것

1 미디어운동/문화연구, dongwon@riseup.net 이 문서는 <문화과학> 60호(2009년 겨울)를 위한 초고임.

2 <아이뉴스24> 2009년 11월 4일자.

3 보통 '리눅스'라고 말하지만 그 운영체제 배포판의 구성은 핵심 요소인 리눅스 커널 외에도 그누 소프트웨어들이 다수 포함되어 있다. 자유소프트웨어운동의 역사적 맥락에서도 '그누/리눅스'로 부르는 것이 더 적합하다.

4 자유 소프트웨어(free software), 오픈소스 소프트웨어(OSS; open source software), 자유/오픈소스 소프트웨어(FOSS) 등으로 불리는데, 이 글에서는 열린 개발 방식에 초점을 두면서 분화해 간 오픈소스를 포함하면서도 사회운동으로서의 자유 소프트웨어를 강조하기 위해 '자유 소프트웨어(운동)'으로 통칭한다.

은 지배적 정보 생산양식의 시장 교환, 노동력 상품화, 위계적 노동분업, 생산수단과 생산물의 사유화, 저작권 체제와는 다른, 그와 정반대의 정보 생산과 소유의 새로운 양식을 만들어 왔다는 이야기이다. 자본주의 사회에서 자본주의를 극복하려는 시도들과 정보 공유지를 구축하려는 최근의 노력들도 이내 자본주의 체제 내로 포섭되어왔고 자유소프트웨어운동 역시 예외는 아니지만 여기에는 뭔가 특별한 것이 있다.

1. 정보 공유지에 울타리 치기: 저작권, 마이크로소프트

1960년대 말까지만 하더라도 컴퓨터 하드웨어는 크고 비싸고 드물었기 때문에 소프트웨어의 복제 자체가 쉽지 않았고, 소프트웨어는 하드웨어와 함께 제공되는 것(bundle)이었다. 물론 당시에도 고용된 프로그래밍 노동자들이 작성한 소프트웨어는 기업의 소유였고 그 소스코드에 대해 저작권이 부여되어 있었지만, 컴퓨터를 사용하는 모두가 프로그래머였기 때문에 이들에게 컴퓨터 시스템의 내부 작동 과정은 개방되어 있었다(Lindenschmidt 2004). 아이비엠(IBM)은 소프트웨어를 소스코드와 함께 무료로 배포했고 컴퓨터 이용자에게 기능 향상을 권장했다. 지배적 하드웨어 제조사 입장에서 더 많은 컴퓨터를 판매하기 위한 사업 방식이었다(Moglen 1999).

아이비엠이 대형 컴퓨터인 '메인프레임'(mainframe)의 최대 판매사였지만, 메인프레임의 설계와 제조는 미국전신전화사인 에이티앤티(AT&T)에서 더욱 활발했다. 다만 기업 내부에서만 사용했는데 무엇보다도 1956년의 반독점법으로 전화회사로서 에이티앤티가 컴퓨터 사업에 진출하지 못하게 된 탓이었다 (Moglen 1999; Söderberg 2008: 14). 에이티앤티 산하의 벨연구소 연구원들이 개발한 유닉스(unix) 역시 비공식적인 프로젝트로 만들어져 자유롭게 배포된 것이다. 이는 특정한 하드웨어에 종속되지 않고 어떤 컴퓨터에서나 작동할 수 있는 하나의 운영체제로 기획된 것이었다. 이를 위해 하드웨어에 의존하는 기계언어나 어셈블러가 아닌 보다 일반화된 언어로 프로그램을 작성하는 것이 필요했고, 벨 연구소에서 만들어진 씨(C)는 그에 적합한 프로그래밍 언어였다(Moglen 1999). 소스코드가 포함되어 수정 가능한 상태로 배포되었기 때문에 시설을 갖춘 대규모 연구소나 대학의 연구원이나 학생들 뿐만 아니라 외부의 이용자들이 값싼 컴퓨터에서 유닉스를 개발하는데 대거 참여하면서 자연스럽게 소형 컴퓨터에 적합하도록 디자인될 수 있었다. 그러나 에이티앤티에 대한 반독점법이 1982년에 해제되면서 컴퓨터 사업이 가능해지자 아이비엠은 유닉스의 소유권을 주장하고 나섰다. 이는 1970년대 말부터 개인용 컴퓨터(PC)의 보급이 소프트웨어 소비 시장을 창출하고, 소프트웨어의 보호기간을 15년으로 한 강제적 라이선스를 제안하는 저작권법안이 제출되면서 크게 변모하기 시작한 소프트웨어산업의 성장을 배경으로 한 것이었다(Söderberg 2008: 15).

하드웨어 해커 공동체에서 탄생한 애플(Apple)이 같은 이름의 벤처자본의 상품이 되면서 컴퓨터가 아마추어 기술 공동체를 넘어 대중 소비 시장으로 결정적 일보를 내딛자 가정용 컴퓨터 시장에 주목한 아이비엠은 마이크로소프트의 급조된 운영체제의 공급 계약을 체결하고 1981년에 '아이비엠 - 개인용컴퓨터'(IBM-PC)를 내놓는다. 1980년대 초부터 개인용컴퓨터(PC) 시대가 되면서 하드웨어 제조사의 제품 차별화 전략의 일환으로 하드웨어에 기본으로 포함된 고품질의 소프트웨어는 사라지고, 독립된 소프트웨어 시장을 독점하려는 경쟁이 거세졌다. 그에 따라 소프트웨어의 수정과 향상을 위한 개발자와 이용자 간의 소스코드의 자유로운 공유는 기업의 핵심 자산을 위협하는 일이 돼버렸다. 약탈적 사업 전략으로 독점의 위치를 차지해온 마이크로소프트의 시장 지배력은 전적으로 저작권이 보호해주는 윈도의 소스코드에 대한 배타적 소유권에 의존한다. 그리고 개인용컴퓨터에 대한 마이크로소프트의 마케팅 전략은 최소 기술 이용자에 맞춘다는 것이었고 프로그래머가 아닌 '일반인'에 적합하게 디자인되었다(Moglen 1999). 생산자와

이용자간의 상호작용이 없어지고 이용자가 탈숙련화된 과정은 소프트웨어의 현저한 품질 저하로 이어졌다. 시장 확대에 따라 개인용컴퓨터의 성능과 용량이 신속히 향상되면서 소프트웨어 자체의 결점은 잘 부각되지 않았고, 일반 이용자에게 소스코드가 접근 가능한 지 안 한 지는 점차 큰 차이가 없게 되었다.

독점 소프트웨어는 소스코드를 배포하지 않는 것과 동시에 제한된 조건 하에서 허용된 이용의 권리 형태로 판매된다. 윈도 씨디를 샀다면 우리가 보통 '동의합니다'를 클릭하고 넘어가는 '최종 이용자 이용권 계약'(EULA)에 따라 단지 한 컴퓨터에서만 설치해 사용하는 것이 허용된다. 친구에게 주거나 변경하거나 다시 팔는 것은 금지된다. 나에게 있는 듯 한데 내 것이 아니다. 다른 물리적 상품과 다르게 소프트웨어 구매는 소유가 아니라 그에 대한 배타적 소유권을 갖는 사람으로부터의 임대이고 이용 중에도 계속 통제를 받는다. 저작권이이러한 계약과 통제를 법적으로 보장하고 있다. 애초 저작권은 '복제물을 생산해 판매하는 권리'의 의미가 강했지만 '권리자가 있는 복제물'에 대한 이용허락을 구매해야 하는 이용자에 대한 규제가 더욱 부각되어왔다(Lindenschmidt 2004).

2. 울타리 걷어차기: 공산과 공유의 정보 생산

2.1. 그누/리눅스와 자유소프트웨어운동

기업과 정부가 저작권과 특허권을 개정하며 컴퓨터 프로그램을 사적 소유물로 울타리치면서 정보의 공유와 자유로운 교환의 공유지에서 떠밀려난 해커 공동체는 정치화되었다.[5] 특히 공유되고 공동 생산돼온 유닉스에 대해 에이티앤티가 소유권을 주장하고 나선 것에 분노한 버클리 대학의 연구자들은 애초 에이티앤티에서 배포될 때 있었던 코드들을 지우고 새로 작성하여 '비에스디유닉스'(BSD; Berkley Software Distribution)를 만들고 이용자들이 자유롭게 소스코드에 접근하도록 하였다. 이것이 리눅스보다 기술적으로 월등한 프로젝트였지만 에이티앤티와의 법적 공방은 이에 대한 개발자들의 참여를 위축시켰다. 메사추세츠공과대학 인공지능연구소의 연구원이었던 리차드 스톨만(Richard Stallman) 역시 협력에 기초한 소프트웨어 개발 공동체의 문화가 파괴되는 것에 분노하며 연구소를 나와 1984년 자유소프트웨어재단을 설립하고 그누(GNU; GNU is Not Unix) 선언문을 발표하며 그누 프로젝트를 시작했다. 이전의 소프트웨어가 그랬듯이 자유 소프트웨어는 누구나 소프트웨어와 그 소스코드를 사용, 배포, 수정, 재배포할 수 있는 소프트웨어로 정의되었다. 그누 프로젝트는 운영체제를 비롯한 다양한 응용프로그램을 개발하는 것이었고 자유소프트웨어재단만이 아니라 전세계의 대학과 컴퓨터센터의 참여와 협력으로 이뤄졌다(Moglen 1999). 운영체제의 핵심 커널 개발에 어려움을 겪다가 1991년 핀란드의 리누스 토발즈(Linus B. Tovalds)가 리눅스 커널을 만들고, 1990년대 초반 인터넷의 확산에 힘입어 전세계의 수많은 해커들이 참여하여 향상시키고 배포하면서 그누/리눅스가 완성되었다.

자유 소프트웨어 운영체제인 그누/리눅스가 윈도우에 대항할 만한 성공을 거두어왔다는 것은 큰 의미가 있다. 그누/리눅스는 컴퓨터 시스템을 구동시키는 핵심 운영체제임과 동시에 자유 소프트웨어의 다른 응용프로그램들이 개발되는 플랫폼이자 근간이기 때문이다. 더 나아가 소프트웨어 원래의 생산 및 유통 방

5 자유 소프트웨어 개발자는 보통 해커(hacker) 혹은 프로그래밍 해커라고 불린다. 1990년대 초반부터 허가받지 않는 컴퓨터 침입을 가리키는 크래커(cracker)가 주류 미디어를 통해 허커로 재현되는 동안에는 거의 알려지지 않았지만, 1950년대 미국 메사추세츠공과대학(MIT)의 해커 공동체로부터 시작된 프로그래밍으로서의 해킹은 지난 10년간 그누/리눅스의 성공과 자유소프트웨어운동의 확장으로 다시 부각되어왔다(Jordan 2009). 해킹의 기원과 전개에 대해서는 조동원, "해킹의 문화정치에서 해킹문화운동으로"(<문화과학> 59호, 2009년 가을) 참조.

식을 복원하고 정보재의 고유한 특성을 인위적으로 막는 저작권 체제와 독점 소프트웨어에 맞서 자유로운 정보 생산과 공유의 문화를 만들어온 것이다. 자유소프트웨어재단에서 시작된 그누 프로젝트는 소프트웨어의 복제, 수정, 재배포와 특히 소스코드에 대한 제한들을 철폐하고 운영체제만이 아니라 모든 소프트웨어를 자유소프트웨어로 만들어 독점 소프트웨어를 사라지게 한다는 이상적인 목표를 가진 것이었다(주철민 2000: 252; Studer 2004).

2.2. '생산의 정치'

2.2.1. 소스코드 혹은 생산수단

소프트웨어의 존재 형태는 목적코드(object code)와 소스코드(source code)의 두 가지로 나뉜다. 소스코드는 우리가 원하는 대로 기계가 작동하도록 특정한 문법에 따라 명령어, 변수, 지시어 등을 논리적으로 나열해 적어 놓은 것과 그에 대한 주석이 포함된 사람이 해석하고 작성하는 언어이다. 목적코드는 소스코드의 내용을 기계가 해석하도록 변환한 (기계)언어이다. 따라서 소스코드 안에는 프로그램의 기능과 기본 구조, 알고리즘, 구현 방법 모두가 담겨져 있다. 소스코드 중에서 명령어, 변수, 지시어 등이 목적코드로 변환되어(compiling) 기계와 커뮤니케이션하게 되는 것이라면, 주석은 이를 보고 다른 프로그래머들이 오류 수정과 기능 향상을 쉽게 하도록 돕기 위해 작성된다(Moglen 1999). 소프트웨어를 사용하기 위해서는 목적코드만 있어도 되지만 그 기능을 변경하거나 향상시키려면 소스코드가 있어야 한다. 소스코드에서 목적코드로 변환하기는 쉽지만 그 반대(de-compiling)는 어렵기 때문이다. 그래서 독점 소프트웨어는 목적코드 형태로만 배포하고 소스코드를 숨겨 아무나 자유롭게 변경하지 못하도록 한다. 대부분의 이용자는 소스코드가 필요하지 않아 별 문제를 못느끼지만 이는 주어진 사용 이외의 다른 행위를 방지하는 통제의 결과이다. 목적코드와 소스코드의 분리는 일견 기계와의 커뮤니케이션과 사람과의 커뮤니케이션의 방식을 다르게 구성하여 효율성을 기한 것이지만, 바로 이 특성에 기초하여 정보를 하나의 사적 소유물로 만들 수 있게 된다(홍원범 2008: 24). 기업의 사적 소유물로 된 소프트웨어는 그 이용허락(라이선스)만을 팔면서 독점의 권력을 갖는 반면, 이용자는 이를 탐구하고 수정하고 배포할 수 없다. 관련된 소프트웨어를 개발하여 그 소프트웨어 시장에 진입하려는 기업이 있다면 처음부터 모든 것을 다 개발해야하기 때문에 사회적인 노동력은 낭비된다.

반면, 자유소프트웨어운동의 오픈소스는 이 소스코드의 개방을 말한다. 애초 소프트웨어의 협력 개발과 공유의 관행대로 그러나 그 반대를 강제하는 저작권 체제 하에서 소스코드가 열리자 정보 생산양식의 새로운 특성들이 부상한다. (자유) 소프트웨어의 생산수단은 소스코드, 디지털 복제 기술, 지구적 네트워크이다. 소프트웨어의 소스코드가 열려 있으면 생산의 자원인 소스코드의 열람, 작성, 수정에 누구나 참여할 수 있다. 그 결과, 한 소프트웨어의 소스코드는 다른 소프트웨어의 생산수단이 된다. 기존 것의 기능을 향상시킨 개작물이든 다른 기능을 갖춘 것이든 개발자는 지금까지 공개된 소스코드를 생산의 자원이자 수단으로 활용할 수 있다. 이를 가지고 쉽게 협력할 수 있고 사회적 노동시간을 절약할 수 있다. 열린 소스코드는 이렇게 생산 대상이자 생산 결과물인 동시에 생산수단이 된다. 그 중에서도 소프트웨어 작성을 돕는 소프트웨어가 있다. 생산을 위한 생산수단이 되는 소프트웨어이다. 소스코드 편집기인 이맥스(Emacs), 목적코드로 변환시키는 그누 컴파일러 모음(GCC, GNU Compiler Collection), 다양한 프로그래밍/스크립트 언어 등이 있다. 자유 소프트웨어는 소프트웨어 생산수단을 직접 만들고 개선하면서 이를 이용해 새로운 소프트웨어를 개발하고, 그것은 다시 누군가에게 또 다른 소프트웨어 개발의 생산수단이 된다.

디지털 복제 기술은 원본과 동일한 복제물의 생산을 거의 비용을 들이지 않고 가능하게 한다. 다행스럽게도 모든 컴퓨터로 가능하고 모든 (디지털) 정보재에 보편적이다. 지구적 네트워크로 연결되어 먼 거리에서

도 디지털 복제물의 손실 없는 공유가 가능하다. 지구적 네트워크로서 인터넷은 자유 소프트웨어 개발의 공동 생산수단이자 협력 작업장이다. 메일링 리스트, 포럼(게시판), 실시간 채팅, 그리고 소스코드 자체에서 직접 협업하는 것이 가능한 도구들(CVS[concurrent versions system]와 같은 동시 개발과 판올림 관리 시스템)이 있다. 이를 통해 협업의 규모나 공간의 제약이 없고 고정된 시간의 제약도 없다. 기존 것의 갈래치기(forking)와 새로운 프로젝트의 시작이 늘 가능하다.

소프트웨어는 또한 공장의 자동화된 조립라인 기계를 작동시키는 생산수단이기도 하고 영화나 게임을 즐기기 위한 소비수단이기도 하다(Victor 2003; 2004). 따라서 자유 소프트웨어는 또한 사회적 생산과 소비 활동 전반에 흐르고 있는 (디지털) 정보가 열린 형태를 갖도록 할 수 있다. 그래서 소프트웨어의 열린 소스코드(와 열린 데이터 포맷)은 소프트웨어 자체뿐만 아니라 이를 매개로 생산된 정보가 배타적 접근 방식으로 제한되지 않고 누구나 누릴 수 있는 공유지가 될 수 있도록 한다. 소프트웨어의 생산수단이자 작업장인 디지털 복제와 지구적 네트워크를 위한 기술과 하부구조 자체를 생산하는 수단 또한 소프트웨어이다. 인터넷의 폭발적인 확산은 거의 정확하게 자유소프트웨어운동의 지구적 성장과 일치했는데, 자유 소프트웨어가 없었다면 인터넷은 개방 네트워크로 발전해올 수 없었다(Lindenschmidt 2004). 아파치(Apache; 웹서버 소프트웨어로 2006년에 전세계 서버의 70%), 샌드메일(Sendmail; 이메일 트래픽 관리), 바인드(BIND; 도메인네임을 IP숫자로 변환), 그리고 웹(www)을 위한 통신규약인 티씨피/아이피(TCP/IP), 에이치티티피(http), 에프티피(ftp), 에이치티엠엘(html), 유알아이(URI) 등이 모두 열린 표준 포맷으로 되어 있다. 이렇게 소프트웨어는 정보자본주의 사회의 하부구조를 구성하고 자유 소프트웨어는 그 하부구조가 (정보기술 기업의 마케팅 전략인 '웹2.0'조차 표방하는) 개방, 공유, 협력의 그것으로 구축되도록 한 생산수단이었다.

2.2.2. 생산공동체의 공동체 생산과정

해커 혹은 프로그래밍 해커가 되기 위해 관련 대학의 졸업장이나 시험을 보고 얻는 자격증이 필요하지 않다. 뭔가를 증명하고 지위를 얻어 프로젝트에 참여하는 것이 아니기 때문이다. 대부분의 해커들은 자신의 관심에 따라 인터넷을 뒤지거나 책을 옆에 두고 직접 해보면서 자기학습한다. 무엇보다도 생산하고 협력하는 과정에서 배워나간다. 프로그래밍 능력이 없더라도 자유 소프트웨어를 이용하면서 생기는 다양한 문제, 의문, 새로운 기능 제안, 사용 소감만을 가지고도 생산에 기여할 수 있다. 프로젝트의 일부가 협업적이더라도 1인에 의해 유지되는 프로젝트가 부지기수다(Studer 2004). 설사 혼자 개발하는 프로젝트더라도 그 소프트웨어를 사용하고 그 소스코드를 들여다보며 배우거나 고칠 수도 있도록 하는 완전한 접근의 보장 덕분에 늘 공동체를 지향하며 열려있다. 따라서 전문 개발자들과 수많은 이용자들로 구성된 공동체에 기초해 생산이 이뤄진다. 즉, 생산자는 이용자의 필요와 요구에 따라 개발(기획, 코드 작성, 오류 수정, 기능 향상)해 나가는 이용자 중심 생산방식이기 때문에 이용자는 생산자의 범주로 포괄된다. 자유 소프트웨어 생산공동체에서는 돈벌이와 같은 유인효과(인센티브) 없이 생산물의 사용가치가 생산의 목표이고 그 자체가 목적인 노동이 행해진다. 이러한 해커들의 노동은 소외되지 않은 노동이고 상품화되지 않은 노동이다. 자신이 생산한 소프트웨어의 판매 그리고 자신의 노동력의 판매를 통해 살아가야하는 노동이 아니라 쓸모있는 소프트웨어를 생산하는데 참여하는 즐거움으로 하는 노동이기 때문이다.

생산 공동체에서 벌어지는 토론과 논쟁, 커뮤니케이션 과정이 곧 생산과정이다. 조단(Jordan 2009)은 해커들이 종종 '돌아가냐?'라고 말할 때처럼 해킹에는 사회적이고 기술적인 논쟁의 즉각적인 해결 가능성을 가진 특정 대상이 있다는 점에 주목한다. 소프트웨어 프로그램인 그 대상은 명령을 실행하고 무엇이든 그 다음에 발생하는 것으로부터 어떤 결론이 도출됨으로써 논쟁을 일단락짓는 방식으로 작업이 진전된다는 것이다. 자유 소프트웨어 프로젝트가 한 회사에 고용된 것도 아니고 언제든지 참여 안 할 자유가 있는

조건에서도 성공적으로 전개되는 것은 바로 이러한 구체적인 해결 과제이자 대상이 있기 때문이다. 해커들이 코드 작성을 하면서 '가려운 데를 긁는다'고 표현하는 것은 자유 소프트웨어 프로젝트에서 노동이 다른 걸 위한 수단이라기보다 그 자체로 목적임을 드러내는 또 하나의 지표이고(Dafermos & Söderberg 2009: 59), 이것은 시원하게 긁어줄 도구로서 생산수단이 개발자와 이용자에게 소유되어 있기 때문에 가능하다.

이러한 생산과정에 따른 자유 소프트웨어의 생산방식은 우선, 협력에 의한 병렬 개발 방식이 두드러진다. 협력의 방식으로서 병렬 개발은 모듈화된 작업의 분담을 말한다. 모듈 체계는 여러 사람들이 프로젝트의 여러 구성 요소의 일부에 대해 독립적으로 작업하면서도 전체를 만들어나갈 수 있도록 한다. 모듈화된 개발 방식의 디자인은 중앙집중적 조정의 필요성을 감소시키고 평행한 생산을 가능하게 한다. 1991년 리눅스 커널이 개발될 때부터 이 방식이 사용되었는데, 리눅스 커널의 평행한 개발 구조는 생산물의 여러 판본의 평행한 출시를 가능하게 하여 1994년 4월 리눅스 커널 1.1판부터 안정화된 판본과 개발 판본으로 나눠졌다(Dafermos & Söderberg 2009: 61). 그러다보니 완벽하게 작업된 것을 공개할 필요없이 작업 과정에서 부딪힌 문제를 포함해 공개함으로써 수많은 개발자들이 달라붙어 이를 함께 해결해 나간다. 자유 소프트웨어의 열린 협력 방식이 갖는 장점은 에릭 레이몬드(Eric Raymond)가 "보고 있는 눈이 충분히 많으면 찾지 못할 오류(버그)는 없다"라고 한 말로 잘 표현된다(주철민 2000: 258-9). 리눅스 커널만이 아니라 자유 소프트웨어 개발 전반에 이러한 모듈화된 개발 방식이 활용되는데, 이는 탈중심적이고 협력적인 소프트웨어 프로젝트를 수행하는데 기술적으로 우월한 것임을 증명하면서 해커 공동체의 의의와 가치를 강화하는 생산방식이 되어왔다(Dafermos & Söderberg 2009: 62). 또한, 생산수단이자 작업장인 지구적 네트워크 덕분으로 그에 참여하는 개발자의 수에 제한이 없다. 인터넷에 접속할 수 있으면 누구나 개발 과정에 유기적으로 참여할 수 있기 때문에 어떤 프로젝트에는 수천 명의 개발자들이 동시에 참여하여 엄청난 속도로 개발이 진행된다. 적어도 하루에 한번 이상 리눅스의 새로운 커널이 발표되기도 했다(주철민 2000: 258). 새로 개발된 자유 소프트웨어가 어느 정도 쓰는 사람들이 생겨나고 불어나면 이용자들의 구체적인 필요와 요구에 대해 즉각적으로 반응하고 상호 되먹임하는 공동체가 형성되면서 삽시간에 진화한다.

수많은 참여자들이 있더라도 상대적으로 적은 핵심 개발자들이 주로 개발 작업을 맡아 해결한다. 아파치(Apache), 모질라(Mozilla), 비에스디(BSD) 등의 대규모 프로젝트에서 상위 10위의 기여자들이 프로젝트 작업의 15%, 상위 30위는 30%를 맡아 작업하고 있다(Dafermos & Söderberg 2009: 63). 이러한 불균등 노동 분담의 양상은 일반적인 개발 과정의 자연스러운 결과로 나타나는 것이다. 그러면서 해커 공동체는 능력 위주 사회의 모습을 강하게 띤다. 어떤 이는 기술적 능력이나 지도력의 발휘에서 다른 이들보다 더 많은 영향력을 갖는다. 그렇더라도 프로젝트에 참여하는 구성원들 모두와 '대략의 합의'(rough consensus)를 거치며 문제를 해결해 나가는 자체적인 거버넌스 방식을 마련하고 있다. 위계적 권위가 존재하더라도 그것을 따를 의무가 없다. 참여자들은 프로젝트 유지자의 기능을 필요로 하고 유지자는 참여자들이 필요하기 때문에 상호의존성에 기초한다(Merten 2007). 문제가 커지고 정 해결이 안 된다면 개발자들은 '퇴장함으로써 불만을 표시하는' 자유를 행사한다. 자유 소프트웨어의 법적 코드(GPL)가 보장하는 것이 바로 기존의 것에서 새로운 것을 갈래치고 나가는 권리이다(Dafermos & Söderberg 2009: 61).

2.3. 사적 '소유의 종말'

2.3.1. GPL: '모든 권리는 뒤집어진다'

자유 소프트웨어의 이용허락은 뭐든지 하도록 허용된다. 더 많은 복제와 배포가 장려되고 내가 원하는 기능을 (친구의 도움으로) 추가할 수도 있고, 더 좋은 기능들을 집어넣고 팔아도 된다. 이와 같이 '자유'의 보

장은 아무도 자신의 소유물을 사용하는데 어떠한 제3자에 통제를 받지 않도록 하는 것이다. 그누 일반공중이용허락문서(GPL; general public license 다음부터 GPL로 줄임)는 소프트웨어를 사용하고, 공유하고, 공부하고, 향상시킬 수 있는 이용자의 자유를 보장하는 자유 소프트웨어의 이용허락이자 법적 코드이다.[6] 프로그래밍 해킹이 자유롭게 소스코드를 (재)작성하는 것이라면 GPL은 저작권의 법적 코드에 대한 해킹이라고 할 수 있다. 법적 권리로서 '주어진' 소유 방식이 아니라 생산공동체 안팎의 사회적 상호작용 속에서 규정되고 정의되는 정보 생산물의 소유 방식이다.

GPL의 '악명높은' 규정은 GPL 달고 배포된 자유 소프트웨어의 소스코드로부터 단 한 줄이라도 가져가 새로운 소프트웨어를 작성했다면 그 전체 소프트웨어에 GPL을 달아야 한다는 점이다. 이를 어기면 기존 저작권법 위반이 된다. 저작권에 의존하면서도 공동 소유권 체계를 기입해 넣은 것이다(Dafermos & Söderberg 2009: 59). GPL의 별칭이 바로 널리 알려진 카피레프트(copyleft)이다. 저작권이 '모든 권리는 보호된다'(all rights reserved)라면, 카피레프트는 '모든 [저작]권리는 뒤집어진다'(all rights reversed)이다. 카피레프트는 저작권에 의존하여 자유 소프트웨어의 자유 원리를 법적으로 표현하고 보장하기 위한 법률적 수단이다. 배제의 권리에 기초해 배포의 권리를 구축하여 제약을 부가하는 대신 다른 사람에게 권리를 부여하는 식이다(Jordan 2009; Vanheuverswyn 2007). 그러나 이것이 가하는 유일한 제약이 있는데 그것은 더 이상 아무런 제약을 가할 수 없다는 제약이다. 상업적 배포를 금지하는 것이 아니다. 다만 그 최초 저작자를 포함해 아무도 소스코드를 독점할 수 없다는 것을 확증하는 것이다. 정보(재)의 속성에 부합하는 자연스러운 일이다. 먹으면 없어지는 한 조각의 빵과 다르게 소프트웨어와 같은 정보는 무한히 복제될 수 있는 빵이기 때문이다. 비경합적일 뿐만 아니라 반경합성을 띠고 사적 소유를 통한 배제가 아니라 포괄을 통해 더 많은 가치가 창출되는 협력재의 속성을 가지고 있기 때문이다.[7] 그래서 저작권이 권리를 부정하면서 희소성을 창조한다면 카피레프트는 권리를 부여하면서 풍요를 창조한다(Merten 2007).

2.3.2. 공유지의 유지와 확장

개발자들에게 GPL이 갖는 의미는 빼앗긴 자들의 이름이 되찾아진 것이었다. 리눅스 커널이 배포될 때 참여자 모두가 새로운 판올림의 기여자 명단(credits) 파일에 언급되면서 참여자들은 그들의 집단 노동의 산물로 그들 자신의 것으로 받아들일 수 있었다. 프로젝트에 기여한 프로그래머들에게 그들의 작업이 공동

6 GPL은 1989년에 스톨만이 작성했다. 당시 주로 사용되던 유닉스 운영체제를 위한 이맥스(Emacs, 소스코드를 편집하는 응용프로그램)는 제임스 고슬링(James Gosling)이 작성한 것이었다. 고슬링은 처음에는 그 소스코드를 아무런 제약없이 배포했고 스톨만은 이를 이용해 그누 이맥스(GNU Emacs)에 통합시켰다. 그런데 고슬링이 마음을 바꿔 그 저작권을 유니프레스(UniPress)라는 회사에 팔자 스톨만는 이 회사로부터 그 소스코드를 사용하지 말라는 통지를 받았다. 이런 일을 겪은 후 스톨만은 GPL을 만들었다(Söderberg 2008: 20). 1991년에 토발즈가 리눅스 커널에 처음에는 독점적 라이선스를 달았다가 이내 GPL을 채택하고 나서 전세계의 개발자들과 함께 개발해 나갔다.

7 비경합성(non-rivalry)은 재화의 동시적 소비가 서로에게 독립적이다. 외부적 강제가 아니라면 희소하지 않은 재화들이 이에 해당한다. 『오픈소스의 성공』(*The success of open source.* Harvard University Press. 2004)을 쓴 스티브 웨버(Steve Weber)가 사용한 반경합성(anti-rivalry)은 흔히 전화와 같이 네트워크 효과를 갖는 재화에서 나타나는데 수많은 사람들에 의해 소비될 때 더욱 유용해지는 것이다. 자유 소프트웨어는 반경합성을 갖는다. 더 나아가 자유 소프트웨어는 네가 가지면 내가 가지지 못하게 되어 배제성을 띠게 되는 것이 아니라 반대로 포괄(inclusion)을 해야 더 유용해지는 것이기 때문에 협력재(collaborative goods)이기도 하다. 협력재는 마크 쿠퍼(Mark Cooper)가 사용한 개념인데 반경합성을 띠면서 포괄성을 띠는 재화로서 일정한 조건 하의 정보재들이 해당한다("From Wifi to Wikis and Open Source: Political Economy of Collaborative Production in Digital Information Age." *Journal on Telecommunications and High Technology Law* 5(1). 2006).

체에 아무 것도 되돌려주는 것이 없는 기업들에 의해 착취되는 게 아니라 모두에게 이득을 주고 자유로운 상태로 남아있을 것이라는 확신을 준 것이다(Vanheuverswyn 2007). GPL은 그누/리눅스 개발 과정에 결정적이었다.

자유 소프트웨어의 소유 방식은 라이선스를 통해 드러나고 라이선스는 거버넌스의 문제이다. 자유 소프트웨어와 자유 소프트웨어의 소스코드 일부로 만들어진 소프트웨어는 계속 자유 소프트웨어여야 한다는 유일한 제약은 이를 가져다가 폐쇄된 소스코드의 독점 소프트웨어가 되는 것을 막는다. 공유지의 사유화를 막는 기능을 하는 것이다. 자유 소프트웨어가 기존의 생산 체계 안에 있으면서도 지속 가능한 대안이 되도록 하기 위해 바로 그 체계를 뒤집어 사용하는 것이다. 자유 소프트웨어가 제공하는 것은 소프트웨어만이 아니라 소프트웨어가 자유롭게 제공된다는 사실 자체에서 연원하는 타자의 자유에 대한 긍정이다. 재배포의 자유는 그 자유와 그에 대한 긍정의 관계를 또 다른 이에게 전하는 것을 독려한다(Studer 2004). 공유지의 확장은 이렇게 이루어진다.

전체적으로 볼 때 자유 소프트웨어 생산공동체는 무엇을 어떻게 생산할 것인가에 대해 국가와 시장의 매개 없이 스스로 결정하는 '생산의 정치'를 실천하면서 자율적 거버넌스와 공유지의 확장 과정을 통해 유지된다. 자유 소프트웨어에서 자유와 개방의 의미는 모든 이가 기여할 수 있고(내적 개방성) 모든 이가 결과물을 이용할 수 있다(외적 개방성)는 점에 있다. 생산 과정의 내적 개방성은 외적 개방성의 전제이고 반대도 마찬가지이다(Merten 2008). 즉, 생산공동체 내의 생산의 민주화와 전체 사회로의 분배의 민주화가 되먹임 순환하며 서로 엮이고 서로 지탱하는 것이다. 이와 같이 자유 소프트웨어는 단순한 취미나 재미('linux for fun')의 문제를 넘어선다. 현대적인 생산수단을 통해 전문가와 일반 이용자의 국적 없는 협력으로 고도의 사회적 유용성을 갖는 현대적인 생산물이 생산되고 되기 때문이다(Merten 2007). 독점 소프트웨어와 자유 소프트웨어라는 생산과 배포 방식의 대립, 그것이 특히 마이크로소프트의 윈도에 대해 전세계 수많은 해커들과 이용자들이 만드는 그누/리눅스로, 모든 컴퓨터에서 사용하는 핵심 소프트웨어인 운영체제에서 경쟁하고 있다는 것은 의미심장하다. 자유 소프트웨어는 시장 교환을 통하지 않고 (교환)가치가 없고 상품이 아니기 때문에 이는 상품들 간의 경쟁이 아니라 상품 일반과의 경쟁이다.

3. 자본의 기생과 (재)종획 ...

마이크로소프트는 자유 소프트웨어 공동체에 대한 색깔 시비를 벌여왔다. 빌 게이츠(Bill Gates)와 현재의 최고경영자인 스티브 발머(Steve Ballmer)는 그누/리눅스 해커들을 공산주의자들이라고 '비난'했고, '리눅스는 지적재산권 체제 내에 기생하여 자신이 접촉하는 모든 것에 달라붙는 암적 존재'라고 말한 바 있다(Friedman 2003).[8] 그러나 마이크로소프트와 그에 경쟁하는 컴퓨터 기업들이 오히려 자유 소프트웨어 공유지에 울타리를 치거나 기생하면서 (독점 혹은 잉여)이윤을 챙겨왔다. 우선, 마이크로소프트는 자신의 시장 독점을 위협하는 기술에 대해 '퍼드'(FUD; Fear, Uncertainty, and Doubt)라는 마케팅 캠페인을 벌여왔는데 마이크로소프트 제품의 '규범'을 벗어나면 공포, 불확실성, 불신에 휩싸인다는 으름장이다 (Lindenschmidt 2004). 이는 특히 자유 소프트웨어에 대한 공격의 수사로 사용되어왔다. 색깔 시비나 암적

8 공산주의자인 해커는 별로 없고 대부분의 해커는 자유주의자에 가깝다. 오픈소스를 자유 시장의 승리로 찬양하는 레이몬드를 비롯한 자유주의적 해커들은 국가 권력에 회의적인만큼 기업 권력을 문제삼지는 않으며 마이크로소프트에 대한 우려는 자유 시장을 방해하는 독점 기업이기 때문이다. 스톨만 역시 해커문화의 고유한 자유주의적 가치로부터 출발했는데 레이몬드가 기업 자유주의자라면 스톨만은 급진적 혹은 좌파적 자유주의자 정도 된다 (Friedman 2003).

존재라는 비난을 들먹이며 GPL 소프트웨어는 소프트웨어 산업의 생태계를 파괴하는 바이러스이기 때문에 각국 정부가 이를 채택하지 말 것을 요구하며 전세계적인 로비를 펼쳐왔다. 마이크로소프트 입장에서는 으레 해온대로 자유 소프트웨어를 인수하거나 금융 수단으로 통제하는 방법이 먹히지 않기 때문에 이데올로기적 공세나 법적인 수단을 동원해 온 것이다(Studer 2004). '디지털권리관리'(DRM, Digital Rights Management)를 도입한 1998년의 디지털천년저작권법(DMCA)처럼 마이크로소프트는 저작권의 최대 이해관계자들과 손잡고 기술과 미디어를 통제하거나 접근 자체를 차단하는 식으로 견고한 울타리를 치고 울타리를 걷어차면 처벌을 받도록 하는 장치들을 만들어왔다.[9]

자유 소프트웨어가 마이크로소프트의 독점 구조에 그 만큼 위협이 되고 있는 것인데, 마이크로소프트와 경쟁하려는 컴퓨터 기업들의 입장에서 자유 소프트웨어는 무료 노동의 원천이고 상대적 잉여가치의 출처가 된다. 토발즈나 레이몬드와 같은 오픈소스 소프트웨어 주창자들과 그 기업들이 해커 윤리를 사업 모델로 만드는데 앞장서왔다. 자유 소프트웨어의 정치적 이데올로기를 걷어내고 열린 소프트웨어 개발에 대한 기업들의 투자를 이끌어내기 위해 1998년 오에스아이(OSI; Open Source Initiative)가 조직되었다. 때를 같이 하여 마이크로소프트의 웹브라우저인 익스플로러에 밀려 어려움을 겪고 있던 넷스케이프사가 자신의 웹브라우저의 소스코드를 공개하는 결정을 내렸고 아이비엠은 아파치와 그누/리눅스에 투자했고 썬, 오라클, 컴팩, 델, 휴렛패커드, 인텔 등의 하드웨어 업체들도 오픈소스 소프트웨어를 지원하고 나섰다. 또한, 오픈소스 소프트웨어를 패키지하고 맞춤 서비스를 제공하며 돈벌이를 하는 레드햇, 노벨, 마이에스큐엘 등의 기업들이 창업되었다.

자유 소프트웨어 프로젝트에 참여하는 적지 않은 해커들이 장기적으로 이런 기업들에서 일자리를 얻는 것을 기대하기도 하고, 이미 관련 업계에서 일하면서 자유 소프트웨어 프로젝트에 참여하는 경우도 허다하다. 생계 문제의 해결은 카피레프트 모델이 해결해주는 것이 아니기 때문이다. 혹은 생계 걱정이 없는 조건에 있는 사람들이 자유롭게 참여한다. 많은 해커들이 산업화된 나라들에 사는 중간(혹은 상층) 계급 출신이고 백인이고 남성이다. 해킹의 자발적 노동과 무상 증여의 문화는 생산자들의 물질적 생계를 보증하는 재화로까지 (아직) 일반화되지 않았고, 이들은 시장 세계에 의해 제공된 사회적 노동과 수익에 의존하고 있다(Victor 2003). 사실상 오늘날 정보자본주의 사회에서 정보기술 산업과 프로그래머라는 선호된 위치는 그 반대의 위치에 있는 다른 노동자들의 희생에 의존함으로써 유지된다. 컴퓨터 기술과 정보기술 산업은 노조의 약화, 노동 시장의 유연화, 노동력의 탈숙련화를 통해 대부분의 사람들이 겪고 있는 자본주의의 신자유주의적 개혁의 기술적 기반이 되어왔기 때문이다(Dafermos & Söderberg 2009: 67). 특히, 저작권 체제를 통한 정보 사유화와 지식 재산화의 울타리에서 배제되는 문화산업(혹은 창조산업)의 창작 노동자들이나 신자유주의적 교육제도 내의 지식 노동자들은 새로운 정보 기술로 인해 탈숙련화되고 저임금을 찾는 작업장 이전이나 아웃소싱으로 불안정 노동이나 실업의 고통을 당하고 있다(Ross 2006: 755). 그런 차원에서, 자유 소프트웨어운동은 지금까지 자유시장주의자인 로렌스 레식과 그 동료들이 이끄는 정보 소비자 중심의 저작권 개혁운동에서 주로 공명해왔지만, 해커들 스스로의 이데올로기와 무관하게 그들의 직접 생산과 공유의 정치적 함의를 고려한다면 이들 정보 프롤레타리아트(운동)와의 수평적 연대가 더 잘 어울린다.

---

9 미디어 콘텐츠의 소비에 '디지털권리관리'가 의무화되면 그렇게 '보호'된 콘텐츠는 엠피쓰리 재생기와 같은 하드웨어와 특정한 독점 소프트웨어에서만 재생되도록 하여 원천적으로 자유 소프트웨어를 배제시킬 수 있게 된다. 이러한 '보호'를 회피하는 해킹이나 기술은 공정한 저작물 이용이더라도 범죄로 규정되었다. 해커들은 이를 '디지털 제약관리'(Digital Restriction Management)라고 부른다. 덧붙여 '신뢰 컴퓨팅'(Trusted Computing) 기술 역시 보안을 명목으로 해서 네트워크를 통해 내 컴퓨터(하드웨어)가 외부에서 통제되는 방식이다. 이용자의 의지와 무관하게 승인되지 않은 프로그램은 자동으로 실행되지 않게 된다.

자본의 기생과 종획에도 불구하고 자유 소프트웨어운동은 여전히 비-자본의 영역들과 반-자본의 투쟁에 유용한 것으로 남아 있다. 지난 몇 년 간 자유 소프트웨어운동은 소프트웨어만이 아니라 그 이상의 영역들로 확장되어왔다: 위키백과, 학술 지식에 대한 열린접근(open access), 자유 하드웨어, 열린 전자제품, 자유 자동차 등등. 자유 소프트웨어 기업들이 성장하고 자본의 울타리치기가 계속 되고 있지만 자본주의를 넘어서는 새로운 사회의 싹들이 여기저기서 생겨나고 있다. 정보의 사유화만이 아니라 모든 사유화의 울타리를 걷어차고 나가는 투쟁과 연대가 이 싹들의 운명을 결정할 장소들에서...

참고문헌

주철민. 2000. "인터넷은 자유다 - 자유소프트웨어 운동." 홍성태·오병일 외, *『디지털은 자유다: 인터넷과 지적재산권의 충돌』*. 이후.

홍원범. 2008. 오픈소스 소프트웨어의 인격적 소유 방식: 공유와 사유의 이분법을 넘어서. 서울대학교 대학원: 인류학과 석사학위논문

Dafermos, G. & Johan Söderberg. 2009. The hacker movement as a continuation of labour struggle. *Capital & Class*, (97).

Friedman, T. 2003. The Politics of Linux. 김영식 옮김. http://blog.jinbo.net/yskim/?cid=3&pid=14 (2009년 10월 30일 접속).

Jordan, T. 2009. Hacking and power: Social and technological determinism in the digital age. *First Monday*, *14*(7).

Lindenschmidt, J. W. 2004. From Virtual Commons To Virtual Enclosures: Revolution and Counter-Revolution In The Information Age. *The Commoner* (9), http://www.commoner.org.uk/?p=20 (2009년 10월 24일 접속).

Merten, S. 2007. Oekonux/Introduction. *OekonuxWiki*. http://en.wiki.oekonux.org/Oekonux/Introduction (2009년 10월 26일 접속).

Merten, S. 2008. The (Open Source) World in 2020: A History Based Look into the Future. *Oekonux*. http://www.oekonux.org/texts/TheWorldIn2020.html (2009년 10월 26일 접속).

Moglen, E. 1999. Anarchism Triumphant: Free Software and the Death of Copyright. *First Monday*, *4*(8).

Ross, A. 2006. Technology and Below-the-Line Labor in the Copyfight over Intellectual Property. *American Quarterly*, *58*(3).

Söderberg, J. 2008. *Hacking Capitalism: The Free and Open Source Software*. Routledge.

Studer, M. 2004. Gift and Free Software. *The Commoner* (9). http://www.commoner.org.uk/?p=20 (2009년 10월 24일 접속).

Vanheuverswyn, M. 2007. The problem with the computer industry under capitalism - Free Software the answer? *In Defence of Marxism*. http://www.marxist.com/computer-industry-capitalism-free-software240907.htm (2009년 10월 26일 접속).

Victor, R. 2003. Free Software and Market Relations. *Oekonux*. http://www.oekonux.org/texts/marketrelations.html (2009년 10월 26일 접속).

Victor, R. 2004. Marxism and free-software (was: [ox-en] Conference documentation / Konferenzdokumentation). *Oekonux*. http://www.oekonux.org/list-en/archive/msg02687.html (2009년 10월 26일 접속).