The Unix Hardware Buyer HOWTO

Eric Raymond

esr@thyrsus.com

Revision History

Revision 1.0	6 February 2001	Revised by: esr

Initial revision; but see the history in the introduction.

This is your one−stop resource for information about how to buy and configure Intel hardware for cheap, powerful Unix systems.

Table of Contents

Table of Contents

Table of Contents

1. Introduction

1.1. Purpose and History

The purpose of this document is to give you the background information you need to be a savvy buyer of Intel hardware for running Unix. It is aimed especially at hackers and others with the technical skills and confidence to go to the mail−order channel, but contains plenty of useful advice for people buying store−front retail.

This document is maintained and periodically updated as a service to the net by Eric S. Raymond, who began it for the very best self−interested reason that he was in the market and didn't believe in plonking down several grand without doing his homework first (no, I don't get paid for this, though I have had a bunch of free software and hardware dumped on me as a result of it!). Corrections, updates, and all pertinent information are welcomed at esr@snark.thyrsus.com. The editorial `we' reflects the generous contributions of many savvy Internetters.

If you email me questions that address gaps in the FAQ material, you will probably get a reply that says ``Sorry, everything I know about this topic is in the HOWTO''. If you find out the *answer* to such a question, please share it with me for the HOWTO, so everyone can benefit.

If you end up buying something based on information from this HOWTO, please do yourself and the net a favor; make a point of telling the vendor ``The HOWTO sent me'' or some equivalent. If we can show vendors that this HOWTO influences a lot of purchasing decisions, we get leverage to change some things that need changing.

Note that in December 1996 I published an introductory article on building and tuning Linux systems summarizing much of the material in this HOWTO. It's available here.

This document actually dates back to 1992, when it was known as the ``PC−Clone Unix Hardware Buyer's Guide''; this was before Linux took over my world :−). Before that, portions of it were part of a Unix Buyer's Guide that I maintained back in the 1980s on USENET.

1.2. New versions of this document

New versions of the Unix Hardware Buyer HOWTO will be periodically posted to comp.os.linux.help and news:comp.os.linux.announce and news.answers. They will also be uploaded to various Linux WWW and FTP sites, including the LDP home page.

You can view the latest version of this on the World Wide Web via the URL http://www.linuxdoc.org/HOWTO/Unix−Hardware−Buyer−HOWTO/.

1.3. Feedback and corrections

If you have questions or comments about this document, please feel free to mail Eric S. Raymond, at esr@thyrsus.com. I welcome any suggestions or criticisms. If you find a mistake with this document, please let me know so I can correct it in the next version. Thanks.

1.4. Related resources

You may also want to look at the read the Hardware−HOWTO. It lists hardware known to be compatible with Linux, and hardware known to be incompatible.

1.5. A discreet plug

I finally found a vendor who consistently lives up to my standards of quality and expertise. In fact, I liked the company so much that I accepted a seat on their board of directors. They're VA Linux Systems, the leading vendor of pre−configured Linux machines. They're not the cheapest (quality and careful engineering costs money) but they're the best. And they're very active in supporting the Linux community.

2. Overview of the Market

The central fact about clone hardware that conditions every aspect of buying it is this: more than anywhere else in the industry, de−facto hardware standards have created a commodity market with low entry barriers, lots of competitive pressure, and volume high enough to amortize a *lot* of development on the cheap.

The result is that this hardware gives you lots of bang−per−buck, and it's getting both cheaper and better all the time. Furthermore, margins are thin enough that vendors have to be lean, hungry, and *very* responsive to the market to survive. You can take advantage of this, but it does mean that much of the info in the rest of this document will be stale in three months and completely obsolete in six.

One good general piece of advice is that you should avoid both the highest−end new−technology systems (those not yet shipping in volume) and the very cheapest systems put out by vendors competing primarily on price. The problem with the high end is that it usually carries a hefty ``prestige'' price premium, and may be a bit less reliable on average because the technology hasn't been through a lot of test/improve cycles. The problem with the low end is that price−cutters sometimes settle for marginal components. Unix works your hardware more efficiently than DOS or Windows, so it is more sensitive to hardware flakiness, which means cut−price systems that might deliver consistently for DOS/Windows lemmings can come around and bite you. Use a little care, and spend the $200−$300 to stay out of the basement. The avoided time and hassles will be worth it.

The last point deserves a little amplification. In the PC world, there's a lot of ``if it doesn't fail, it's OK''. It is common to ignore normal engineering tolerances (allowances for variations in components, temperature, voltage margins, and the like) and to assume that anything which doesn't fail outright must work. Watch out! As a historical example, the ISA bus was originally designed for 6 MHz. IBM later updated that to 8 MHz, and that's as much of a standard as there is, yet there were motherboards that will let you (try to!) run it at 12 MHz, 50% over spec. Some cards were actually designed to work at that speed with proper tolerances. Others might work...or they might flake out when they get warm. Any systems vendor above the fly−by−night level is going to shoot for a little more reliability than this, burning in systems and (often) doing at least a token system test with some kind of Unix (usually SCO XENIX). Pay the few extra bucks it costs to deal with a more careful vendor.

The happy bottom line is this: at July 2000 direct−mail prices, you can expect to get an AMD K6 or Pentium III 450 system with 64MB of memory, 6gig EIDE hard disk, 3.5 floppy, 101−key keyboard, 32X CD−ROM drive, sound card & speakers, SuperVGA−compatible 17'' monitor, 56KB modem, and a decent AGP video card for $1000 or even less. This is a more than reasonable Unix and X machine.

I put together the first version of this guide around 1992; Unix−capable systems are now five to ten times cheaper than they were then. At today's prices, building your own system from parts no longer makes much sense at all −− so this HOWTO is now more oriented towards helping you configure a whole system from a single vendor.

3. Buying the Basics

In this section, we cover things to look out for that are more or less independent of price−performance tradeoffs, part of your minimum system for running Unix.

Issues like your choice of disk, processor, and bus (where there is a strong tradeoff between price and capability) are covered in the section on performance tuning.

3.1. Getting Down to Cases

Cases are just bent metal. It doesn't much matter who makes those, as long as they're above an easy minimum quality (on some *really* cheap ones, cards fail to line up nicely with the slots, drive bays don't align with the access cutouts, or the motherboard is ill−supported and can ground out against the chassis). If you're fussy about RFI (Radio−Frequency Interference), it's worth finding out whether the plastic parts of the case have conductive coating on the inside; that will cut down emissions significantly, but a few cheap cases omit it.

Should you buy a desktop or tower case? Our advice is go with tower unless you're building a no−expansions personal system and expect to be using the floppies a lot. Many vendors charge nothing extra for a tower case and the absolute maximum premium I've seen is $100. What you get for that is less desktop clutter, more and bigger bays for expansion, and often (perhaps most importantly) a beefed−up power−supply and fan. Putting the box and its fan under a table is good for maybe 5db off the effective noise level, too. Airflow is also an issue; if the peripheral bays are less cramped, you get better cooling. Be prepared to buy extension cables for your keyboard and monitor, though; vendors almost never include enough flex.

The airflow thing is a good argument for a full− or mid−tower rather than the `baby tower' cases some vendors offer. However, smaller towers are getting more attractive as boards and devices shrink and more functions migrate onto the motherboard. A state of the art system, with all 3" disks, 200W power supply, half−size motherboard, on−board IDE and 64meg of RAM sockets, and half−sized expansion cards, will fit into a baby or midized tower with ample room for expansion; and the whole thing will fit under a desk and make less noise than a classic tower.

For users with really heavy expansibility requirements, rackmount PC cases do exist (ask prospective vendors). Typically a rackmount case will have pretty much the same functionality as an ordinary PC case. But, you can then buy drive racks (complete with power supply), etc. to expand into. Also, you can buy passive backplanes with up to 20 or so slots. You can either put a CPU card in one of the slots, or connect it to an ordinary motherboard through one of the slots.

3.2. Power Supplies and Fans

Power supplies can matter but quality is cheap; give preference to those with a Underwriter's Laboratories rating. There's some controversy over optimum wattage level. On the one hand, you want enough wattage for expansion. On the other, big supplies are noisier, and if you draw too little current for the rating the delivered voltage can become unstable. And the expected wattage load from peripherals is dropping steadily. The big old 300−watt supplies that were designed for running several full−height 5.25" floppies and hard disks are overkill in these days of portable−ready lightweight 3.5" drives. 200 watts is good enough these days, and the new breed of compact 200W supplies is quieter to boot.

About that annoying fan noise, ask if the power−supply fan on a target system has a variable speed motor with thermostatic control; this will cut down on noise tremendously. However, be aware that a thermostatic sensor basically measures the temperature *at the sensor* (typically within the power supply box) and makes sure there is enough airflow to keep the power supply from overheating. The sensor does not know a thing about the temperature in certain hot spots likely to develop in a PC case (CPU, between SIMMs, between drives mounted in vertically adjacent bays).

This can be a problem, because in garden variety tower cases there often isn't enough airflow to cool all components effectively even if a single fan is going at full speed. This is especially true if your computer has lots of add−on cards or hard disks (not much airflow between cards or between drives). Note that the fan in the power supply was basically designed to cool the power supply, not the components in the case. Not providing additional fans is a sign of cheapness. On tower PCs with "expensive" engineering (e.g. HP Vectra, Compaq) one will find one to two extra fans besides the one in the power supply.

So the bottom line is, use thermostatic controls if you can to cut noise. But if you want high reliability, use two or more fans. Modern designs will usually have a small auxilliary fan mounted right over the chip! This is a good feature to ask about in advance.

The noise produced by a fan is not just a function of the speed with which it turns. It also depends on the nature of the airflow produced by the fan blades and the bearings of the rotor. If the blades causes lots of turbulent airflow, the fan produces lots of noise. One brand of fans that, I'm told, has a reputation for being much more silent than others even if going at full throttle is the German manufacturer Pabst. Their fans are offered in US computer magazines.

3.3. Motherboards

Provided you exercise a little prudence and stay out of the price basement, motherboards and BIOS chips don't vary much in quality. There are only six or so major brands of motherboard inside all those cases and they're pretty much interchangeable; brand premiums are low to nonexistent and cost is strictly tied to maximum speed and bus type. Unless you're buying from a "name" outfit like Compaq, Dell, or AST that rolls its own motherboards and BIOSes, there are only four major brands of BIOS chip (AMI, Phoenix, Mylex, Award) and not much to choose between 'em but the look of the self−test screens. One advantage Unix buyers have is that Unixes are built not to rely on the BIOS code (because it can't be used in protected mode without more pain than than it's worth). If your BIOS will boot properly, you're usually going to be OK.

The ``Baby AT'' motherboard form factor is obsolete. It's an ATX world now.

There are still a few potential gotchas to beware of, especially in the cheaper off−brand boards. One is ``shadow RAM'', a trick some boards use for speeding up DOS by copying the ROM contents into RAM at startup. It should be possible to disable this. Also, on a cacheing motherboard, you need to be able to disable cacheing in the memory areas used by expansion cards. Some cheap motherboards fail to pass bus−mastering tests and so are useless for use with a good SCSI interface; on others, the bus gets flaky when its turbo (high−speed) mode is on. Fortunately, these problems aren't common and are becoming less so.

You can avoid both dangerously fossilized hardware and these little gotchas by sticking with a system or motherboard design that's been tested with Unix.

Some other good features to look for in a motherboard include:

- Gold−plated contacts in the expansion slots and RAM sockets. Base−metal contacts tend to grow an oxidation layer which can cause intermittent connection faults that look like bad RAM chips or boards. (This is why, if your hardware starts flaking out, one of the first things to do is jiggle or remove the boards and reseat them, and press down on the RAM chips to reseat them as well −− this may break up the oxidation layer. If this doesn't work, rubbing what contacts you can reach with a soft eraser is a good fast way to remove the oxidation film. Beware, some hard erasers, including many pencil erasers, can strip off the plating, too!)
- The board should be speed−rated as high as your processor, of course. It's good if it's rated higher, so upgrade to a faster processor is just a matter of dropping in the chip and a new crystal.

If you're changing a motherboard, see the Installing a Motherboard page first. This one even has a Linux note.

3.4. Memory

All current PC designs include a cacheing memory controller and some SRAM that combine to produce higher effective speeds. Be sure to read the `Cache Flow' section under performance tuning for more on how cache configuration affects your performance.

Current motherboards use PC100 SDRAM, which comes packaged on 168 pin DIMM modules. SDRAM is a large step forward in memory speed, at 10ns. SDRAM does not need to be installed in pairs. The key words you want to see on the spec sheet are:

- SDRAM DIMM −− the physical module type.
- ECC −− error correcting memory, important for reliability; Unix makes more efficient use of hardware and thus beats on your memory harder than Windows does.
- PC100 −− this is the memory−module specification for current motherboards with a 100Mhz PCI bus.
- 16x72 −− if it says 16x64, the extra 8 bits per 64 needed for ECC are not present.

Also, avoid anything that says `ECC compatible memory'; this is the *opposite* of `memory with ECC'.

For more technical stuff on memory architectures, see The Ultimate Memory Guide maintained by Kingston Technologies.

3.5. Buying a Video Card

Video controllers translate byte values deposited in their video memory by your GUI (usually an X server under Linux) into an analog RGB signal which drives your monitor. The simplest kinds treat their video memory as one big frame buffer, requiring the CPU to do all dot−painting. More sophisticated ``accelerated'' cards offer operations such as BitBlt so your X server can hack the video memory algorithmically. These days almost all cards even at the low end actually have some acceleration features.

Cards are rated by the maximum number of analog signal changes they can produce per second (video bandwidth). Video bandwidth can be used to buy varying combinations of screen resolution and refresh speed, depending on your monitor's capabilities.

Another important variable of video cards is the size of their on−board video RAM. Increased memory lets you run more colors at higher resolutions. For instance, a 1MB card usually will only allow 256 colors at 1024x768 while a 2MB card usually allows at least 16−bit color (a palette of about 65,000 colors). You'll need 4MB of video memory to use 24−bit or ``true'' color (16 million colors) at 1024x768.

The card's video RAM size has no effect on its speed. What does affect speed is the *type* of memory on board. VRAM (Video Random Access Memory) is fast but more expensive; it features a dual−ported design allowing two devices (the CRT controller and the CPU) to access the memory at the same time. DRAM (Dynamic Random Access Memory) is is similar to the RAM used in main memories. It is cheaper, more common, and slower (because the CRT controller and the CPU must take turns accessing the video buffer).

A quick review of monitor standards:

Table 1. Monitor standards

Name	Resolution	Colors	Horizontal Frequency	Vertical Frequency	Notes
MDA	720x350		18.43 KHz	50 Hz	Obsolete
CGA	640x200	2	15.85 KHz	60 Hz	Obsolete
	320x200	4			
EGA	640x350	16	21.80 KHz	60 Hz	Obsolete
VGA	640x480	16	31.50 KHz	60 Hz	
	320x200	256			
VESA VGA	640x480	16	38.86 KHz	72 Hz	
	320x200	256			
VESA SVGA	800x600	16	48.01 KHz	72 Hz	
	640x480	256			
8514/A	1024x768	16	35.20 KHz	43.5 Hz	Obsolete
XGA	64x480	65536	??	??	Obsolete
	1024x768	256			
VESA 1024x768	1024x768	256	56.48 KHz	70 Hz	

The Horizontal and Vertical Frequency columns refer to the monitor scan frequencies.

The vertical frequency is the upper limit of the monitor's flicker rate; 60Hz is minimal for ergonomic comfort, 72Hz is VESA−recommended, and 80Hz is cutting−edge. At resolutions above VGA, horizontal scans take long enough that the monitor may never reach anywhere near the vertical−frequency maximum; how close it gets is a function of the horizontal−scan frequency (higher is better).

For more information on how to avoid the evil screen flicker, see the XFree86 Video Timings HOWTO, a tutorial written by your humble editor and included with the XFree86 distribution.

It's still barely possible to find MDA video boards and monitors out there, but prices for SVGA have collapsed so totally that it's not worth bothering.

XGA is an IBM−proprietary included for completeness, but is vanishingly rare in the clone market. 8514/A is another IBM standard supported by a few graphics accelerator cards. It is interlaced, and thus has a tendency to flicker. The VESA 1024x768 standard makes XGA and 8514/A obsolete.

SVGA or `Super VGA' strictly refers only to 800x600 resolution, but is widely used for 1024x768 and even 1280x1024 resolutions. Standards above 1024x768 are weak and somewhat confused.

These days, most vendors bundle a 14" monitor and super−VGA card with 1024x768 resolution in with their systems. Details to watch are the amount of RAM included (which will affect how much of that maximum resolution and how many colors you actually get), and whether the memory is dual−ported VRAM (slightly more expensive but much faster).

You should check ahead of time which Super VGA chipset the vendor normally ships. Though DOS/Windows doesn't really care, the Unix software that uses it (most notably X servers) will definitely notice the difference. Many commercial implementations of X servers for Intel Unix don't know how to use the SVGA modes of the cheapie Oak and Trident SVGA chipsets, which are the ones most often bundled with systems (XFree86 handles them OK, though). The cards based on the Tseng 4000 chipsets are often bundled or available as as an extra−cost option (usually less than $50 above Oak or Trident), perform better, and are supported by the most implementations of Intel Unix−based X servers. ATI VGAWonder chipset is well−supported and often bundled with multimedia or graphics systems.

More information on video standards is available at the Video Display Standards page of the PC Guide site.

3.6. Selecting a Monitor

3.6.1. What To Look For On The Spec Sheet.

- Dot pitch of 0.28 or smaller on a 12"−15" monitor; 0.30 is acceptable on larger ones, especially 19" to 21" screens (but look extra hard at 0.25 21−inchers like the Viewsonic 21PS or Nokia 445X). Dot pitch is the physical resolution of the screen's phosphor mask. Larger dot pitches mean that small fonts and graphic details will be fuzzy.
- 72Hz or better vertical scan frequency, to cut flicker.
- Non−interlaced display. Interlacing cuts the required scan frequency for a given resolution in half, but makes flicker twice as bad. As a result, interlaced monitors are rapidly disappearing; don't get stuck with one.
- Multisync capability. These days, only the cheapest low−end monitors have fixed horizontal−scan frequencies. If you buy one, it's potluck whether it will work with your present graphics board, let alone your next one. Your monitor is a big investment, and likely to outlive a generation or two of graphics boards. Don't get stuck!
- Does it have a tilt−and−swivel base? Adequate controls, including both horizontal and vertical size and horizontal and vertical centering? A linearity control, a trapezoidal control, and a color−temperature control are all pluses; the last is particularly important if you compose graphics on screen for hardcopy from a printer.
- Is it *color*? If you don't see it in the ad, ask; some lowball outfits will try to palm off so−called "black & white VGA" monitors on you.
- For X use, a 14", .28mm dot pit, non−interlaced 72Mhz monitor at 640x480 resolution is the bare minimum for comfortable use, and that resolution leaves you rather squeezed for screen real estate. 1024x768 is much better. If your budget will stand it at all, a 17" or 20" monitor is a good investment. A 17" monitor is minimum if you're going to go with 1280x1024 resolution.

If you can, buy your monitor from someplace that will let you see the same monitor (the very unit you will walk out the door with, not a different or `demo' unit of the same model) that will be on your system. There's a lot of quality variation (even in "premium" monitor brands) even among monitors of the same make and model.

Another good reason to see before you buy, and carry it home yourself, is that a lot of monitors are vulnerable to bumps. The yoke can get twisted, producing a disconcerting tilt in the screen image.

The Caveat Emptor guide has a good section on evaluating monitor specifications. And there's a database of monitor specs at The Big Old Monitor List.

3.6.2. Eric Buys A Big Monitor: Smart Shopping Tips

In early 1996 the good folks at O'Reilly Associates dropped several $1000 checks on me in relatively quick succession (payment for fast−turnaround technical reviews). I decided to use the money to treat myself to a really good monitor.

This page tells you how I did it. Specific specs and pricing information will date quickly, but the method should still be good years from now.

My existing monitor wasn't bad −− a 17−inch Swan 617 that I could drive at a bit above 1024x768. Still, I yearned for more real estate −− especially vertical real estate, so I could view full PostScript pages using a legible font.

This brings us to our first prescription: *be clear about what you want*. It's easy, and very expensive, to buy more monitor than you'll really use.

I knew I wanted something in the 19−to−21−inch range, with 1280x1024 or higher resolution. I knew this would probably cost me about $2000, and could afford it. I knew I *didn't* need one of the monster projection monitors further upmarket, with screen sizes 24" and up. These will typically cost you $4K or so and are too big for desktop use anyway.

I also knew I didn't need one of the special true−color monitors designed for photo composition, making print separations, and so forth. These creatures (always Trinitrons) have better, denser color than conventional tubes but at a hefty price premium (and usually at some cost in available resolution). If all you're going to do most of the time is 16 or 256−color X screens, you don't need this capability.

Once you've settled on what you need, *gather comparative data*. It was 1996, so I started out by making phone calls to manufacturer 800 numbers. Then I discovered that almost all the manufacturers had Web sites, with technical specs for their monitors on them. Today, you'd go to the Web first.

(This space used to inclide detailed technical data on what I found " model numbers, resolutions, reviews of manufacturer websites, etc." but I've removed it because it's all five years out of data now.)

This wasn't at all a hard call. The ViewSonic 21PS and Nokia 445X stood out from the pack immediately; their combination of high bandwidth with a 21−inch screen size and ultra−fine .25 dot pitch promised better performance than the general run of .28−pitch monitors.

Nor was the choice between the two very hard. ViewSonic's 21PS is $600 less expensive than Nokia's 445X for very similar performance. And, other things being equal, I'd rather buy a monitor from a specialist monitor manufacturer than a general consumer electronics outfit best known for its cellular phones.

So I determined to order a ViewSonic 21PS.

This left me with a second problem. My ATI Mach 32 can't drive a monitor at higher than 1280x1024 resolution and 94MHz bandwidth. So it wouldn't be able to drive the 21PS at 1600x1200. I wound up buying a Mach 64.

The combination worked wonderfully (two years later I discovered that VA Linux Systems buys the same monitor for its high−end systems). The only problem I have with it is that monitor is *way* bright even dialed down to its dimmest setting. You'll need a strong light in the room where you install it. Also, be aware that the only really convenient way to move one of these monster monitors is with a forklift!

3.7. Keyboards

It's important to get a high quality keyboard with good key feel. See the typing−injury FAQ from sci.med.occupational to see what happens if you don't. Carpal tunnel syndrome is no fun for anyone, but it hits hackers particularly hard. Don't be a victim!

Hal Snyder of Mark Williams, Co sent us the following caveat: "We find that about 10% of cheap no−name keyboards do not work in scan code set 3. We are interested in scan code set 3 because only there can you reprogram the keyboard on a per−key basis as to whether keys are make−only, make−break, or autorepeat. It is a big win for international support and for X."

He continues: "Keytronic, Cherry, and Honeywell keyboards, as well as a large number of imports, work fine. My advice is to either buy a respected brand of keyboard, or deal with a vendor who will allow you to return an incompatible keyboard without charge."

Allen Heim <ajh@netscape.com> writes: I'm sold on NMB keyboards (http://www.nmbtech.com/), available from Global Computer (http://www.globalcomputer.com/), or at (800) 8−GLOBAL. Their line of mechanical−switch keyboards now have a lifetime warranty, and I've just ordered my second RT−8200 unit. I don't see it listed on their website currently, but I do see their "Windows 95−enhanced" model, the RT−8200W. It's the same thing, but with extra keys (which may be programmable; think of the Emacs "meta" key−−could be useful).

In the wake of recent publicity about reprtitive−strain injuries, ergonomic keyboards are becoming increasingly common. One that looks promising to your editor (though I haven't yet used it) is the Marquardt MiniErgo MF2, from Marquardt Switches, Inc.; 2711 Route 20 East, Cazanovia NY 13035, phone (315)−655−8050; suggested list price $170, AT−compatible interface). Michael Scott Shappe <mshappe@jeeves.net> sent me a rave review of the Marquardt after having used it about six months.

The MF2 features a conventional QWERTY layout, but with the right and left halves split apart and rotated about 30 degrees towards each other in a shallow V shape. The theory is that being able to angle your arms inward and your elbows out produces a less stressful typing position.

The MF2 has no keypad, but it does have the standard 12 function keys across the top and arrow keys at the point of the V (meant to be thumb−operated).

I have seen and used a device called the Maltron ergonomic keyboard. This keyboard splits the keys into two main groups, each arranged in a dished hollow. Each hand also has easy access to separate thumb pads of nine keys each; there is one numeric and arrow−key pad in the center of the unit, between the right and left−hand groups. Also, the keys in each group do *not* have the alternate−row staggering of the conventional (Scholes) keyboard; this subtle change reduces torsional stress on the fingers and wrists tremendously.

I found the Maltron easy to use, and regretted having to give it back.

For more details on many ergonomic keyboards and typing−injury issues in general, see Dan Wallach's FAQ on repetitive strain injuries and ergonomic input devices, published monthly in news.answers

3.8. Floppy Drives

There's not much to be said about floppy drives. They're cheap, they're generic, and the rise of CD−ROM drives as a cheap distribution medium has made them much less important than formerly.

The media come in two sizes −− 3.5−inch `hard−shell' floppies and 5.25" flexible floppies. The 3.5" variety carries 1.44MB of information; the older, larger 5.25" carries 1.2MB. The 5.25" variety is obsolete, but is still included with some new systems as a sort of vermiform appendix, like the useless ``Turbo'' buttons on many cases.

You'll probably never use floppies for anything but first boot of a new operating system, and can almost certainly do without a 5.25" drive entirely.

Go ahead and settle for cheap Mitsumi and Teac floppy drives. There are no `premium' floppy drives anymore. Nobody bothers. If you're seriously short of expansion−bay space, you can get a combination drive with 5.25 and 3.5" slots on a single spindle.

3.9. Printers

There's one huge gotcha about printers: *Don't Get a `GDI' printer*!

3.9.1. GDI printers: avoid!

Low−end printer manufacturers have been increasingly moving towards `GDI' or `Windows' printers for their laser or LED models. These have the unfortunate characteristic of being, at worst, unusable from Unix; and at best, useable only with reduced resolution (e.g., 300 dpi from a 600 dpi printer).

The problem is that the design of laser printers inherently requires that data move onto the imaging drum at a precisely−controlled rate, and so laser (and LED) printers have traditionally included a CPU of moderate speed and enough RAM to image an entire page, either as a complete bitmap or as ASCII, using a quick font rasterizer to form images on−the−fly. GDI printers, however, offload this responsibility to the computer, and therefore require very specialized drivers that (a) are not available for any Unix, AFAIK; and (b) slow down the computer a lot when printing is underway.

Most GDI printers DO support HP's PCL in a lower resolution, so they can often be used from Unix via Ghostscript, but only in 300 dpi and/or missing some features. They may also be slower in this mode than in their native GDI modes. In order to print at 600 dpi with Ghostscript, a PCL printer must support HP's PCL at level 5e or better, so printer purchasers should look for that in any non−PostScript model, at least at the moment. (Naturally, all of this could change if/when support is added to Ghostscript for more esoteric models; but AFAIK, this is the current set of limitations.)

Finally, I've seen one extra twist on this already−twisted marketplace: The Brother HL−720 is advertised as supporting `PCL 5e for DOS' (or words to that effect). What this means is that it's a GDI model with a DOS driver that takes PCL 5e input and translates it to the printer's native GDI mode. Needless to say, this is useless for Unix.

GDI printers are a bad design even for the DOS lemmings, because they slow the machine down significantly while printing is going on. Like `WinModems', they're a sleazy way for manufacturers to save a few bucks. Our advice is, buy a printer with native Postscript and avoid all this crap.

3.9.2. Non−GDI printers

There really isn't all that much to be said about printers; the market is thoroughly commoditized and printer capabilities pretty much independent of the rest of your hardware. The PC−clone magazines will tell you what you need to know about print quality, speed, features, etc. The business users they feed on are obsessed with all these things.

Most popular printers are supported by GhostScript, and so it's easy to make them do PostScript. If you're buying any letter−quality printer (laser or ink−jet), check to see if it's on GhostScript's supported device list −− otherwise you'll have to pay a premium for Postscript capability! Postscript is still high−end in the MS−DOS market, but it's ubiquitous in the Unix world.

Warning, however: if you're using ghostscript on a non−Postscript printer, printspeed will be slow, especially with a serial printer. A bitmapped 600 dpi page has a *lot* of pixels on it. Further, if you're doing much printing, ghostscript will create enormous spool files. (megabytes/page). At today's prices, paying the $750 or so for Postscript capability makes sense.

If you're buying a printer for home, an inkjet is a good choice because it doesn't use gobs of power and you won't have the toner/ozone/noise/etc mess that you do with a laser. If all you want is plain−ASCII, dot−matrix is cheaper to buy and run.

Inkjets are great in that they're cheap, many of them do color, and there are many kinds which aren't PCL but are understood by Ghostscript anyway. If you print very infrequently (less than weekly, say), you should be careful to buy a printer whose print head gets replaced with every ink cartrige: infrequent use can lead to the drying of the ink, both in the ink cartrige and in the print head. The print heads you don't replace with the cartrige tend to cost nearly as much as the printer (~$200 for an Epson Stylus 800) once the warranty runs out (the third such repair, just after the warranty expired, totalled one informant's Stylus 800). Be careful, check print head replacement costs ahead of time, and run at least a cleaning cycle if you don't actually print anything in a given week. (Conversely, toner starts out dry, and ribbon ink won't evaporate for years...if you truly print only rarely, but neither a dot matrix nor a laser makes sense, consider buying no printer and taking your PostScript files to a copy shop...)

A parallel interface is a cheap way to make your printer print a lot faster than a serial line, and everyone's got a parallel port in their PC.

A few printers for the MS−DOS market require a special controller card and proprietary cable to do PostScript. These require MS−DOS software and typically won't run under Unix at all.

Meanwhile, there are several true 600 dpi lasers that grok PCL 5e, yet cost less than $500 retail. Currently (December 1997) these include the Lexmark Optra E (and E+), the HP 5L (and 5L with suffix, and probably 6L), and the Brother 760. As you can't easily buy a new hard drive smaller than 2 gigabytes, tens of megabytes of spare space in /var/spool should be the accepted norm, rather than a problem, for new systems; I've also noticed that PCL 5e seems to include some amount of compression (probably RLE or font encoding) which works rather well for text, further reducing the spool requirements.

One of our spies says good things about the Canon BJC−240 and 250. He reports they preint well with Ghostscript and are more reliable than Deskjets.

I personally have a LaserJet 6MP, and like it.

3.10. Power Protection

3.10.1. Overview

Finally, I strongly recommend that you buy a power conditioner to protect your hardware. MOV−filtered power bars make nice fuses (they're cheap to replace), but they're not enough.

The technical info in the remainder of this section is edited from material supplied by David E. Wexelblat >dwex@mtgzfs3.att.com<.

There are several levels of power protection available to the home computer user. I break this down into 4 levels; others may have different ways of classifying things. The levels are:

1. Surge Suppressor
2. Line Conditioners
3. Standby Power Supplies
4. Uninterruptible Power Supplies

And here's what they mean:

3.10.2. Surge suppressors

These are basically a fancy fuse between the source and your hardware; they clamp down spikes, but can't fill in a low voltage level or dropout.

This is a bare minimum level of protection that any piece of expensive electronics should have. Note that this applies to more than just AC power; surge suppressors are available for (and should be used on) phone lines, and RS−232 and parallel connections (for use on long lines; generally not needed if the devices is colocated with the computer and all devices are protected from outside sources). Note also that *all* devices connected to your computer need to be protected; if you put a surge suppressor on your computer but not your printer, then a zap on the printer may take out the computer, too.

An important fact about surge suppressors is that *they need to be replaced if they absorb a large surge*. Besides fuses, most suppressors rely on on components called Metal−Oxide Varistors (or MOVs) for spike suppression, which degrade when they take a voltage hit. The problem with cheap suppressors is that they don't tell you when the MOV is cooked, so you can end up with no spike protection and a false sense of security. Better ones have an indicator.

You can buy surge suppressors at any Radio Shack; for better prices, go mail−order through Computer Shopper or some similar magazine. All of these are low−cost devices ($10−50).

3.10.3. Line Conditioners

These devices filter noise out of AC lines. Noise can degrade your power supply and cause it to fail prematurely. They also protect against short voltage dropouts and include surge suppression.

My Tripp−Lite 1200 was typical of the better class of line conditioners; a box with a good big soft−iron transformer and a couple of moby capacitors in it and *no* conductive path between the in and out sides. With one of these, you can laugh at brownouts and electrical storms. You can get for $139 or so by mail order. A fringe benefit of this little beauty is that if you accidentally pull your plug out of the wall you may find you actually have time to re−connect it before the machine notices (I did this once). But a true SPS or UPS is better.

Netter Trey McLendon has good things to say about Zero Surge conditioners. He says: "Our systems at work [...] have been protected for 2.5 years now through many a violent storm...one strike knocked [out] the MOV−type suppressors on a Mac dealer's training setup across the street from us. The Zero Surge just sort of buzzed when the surge came in, with no interruption whatsoever. The basic principle is this: ZS units slow down the surge with a network of passive elements and then sends it back out the neutral line, which is tied to ground _outside at the box_ by code. MOV units shunt the surge to ground _at the computer_, where it leaps across serial ports, network connections, etc. doing its deadly work."

Price vary widely, from $40−400, depending on the power rating and capabilities of the device. Mail−order from a reputable supply house is your best bet. Line conditioners typically *don't* need to be replaced after a surge; check to see if yours includes MOVs.

3.10.4. Standby power supplies (SPSs)

These devices are battery−based emergency power supplies that provide power for your system via an inverter if the power fails. An SPS will generally have all the capabilities of a line conditioner as well.

Note: these devices do not come on line until after the power fails, and have a certain amount of delay (typically some milliseconds) before they come on line. If the capacitors in your power supply are not large enough, the SPS may not cut in soon enough to prevent your computer from seeing the power failure.

Note also that many SPSs are marketed as Uninterruptable Power Supplies (see below). This is incorrect. Any device with a non−zero cutover time cannot be a true UPS. If the ad mentions a cutover time, it's an SPS, and not a UPS.

The price range for these devices (increasing with increasing peak load capacity and with decreasing cutover time) is $200−2000. An SPS will *not* need to be replaced after absorbing a large surge.

3.10.5. Uninterruptable power supplies (UPSs)

These devices provide full−time isolation from the incoming AC line through a transformer of some sort. These devices are on−line at all times, and if the AC line fails, the batteries will cut in. Your devices will see no interruption of their incoming AC. UPSs cost more, and provide more features. They are the ultimate in power protection. Many UPSs have an intelligent interface that will notify a connected device of a power failure, allowing it to shut down cleanly. UPSs also provide the capabilities of a line conditioner. The price range (for devices in the size range for a home computer) are $200−$1500. An UPS will *not* need to be replaced after absorbing a large surge.

Now, given this information, how does one decide what to get? For a system that runs unattended, like most Unix systems, it is best to have a device that provides both power holdover and a power failure signal. Hence, for a Unix system, a UPS or SPS with Unix monitoring software is the best choice.

If the vendor isn't secretive about interface specs, it's fairly simple to write your own daemon to monitor a serial port, and send init a SIGPWR signal when it sees a powerdown notification on the port. Freeware power−monitor demons are available for Linux.

Many UPS/SPS signal ports work by asserting a pin, so that one could use a modem−control serial port on the PC and wire this pin to "Carrier Detect" in order to monitor it. Some, like the APC "SmartUPS" series, actually conduct an ASCII dialog with the host through a serial line in order to accomplish the monitor functions.

Our recommendation for a production Unix environment is a configuration like the following:

1. An on−line UPS or SPS for the computer system. An intelligent interface is mandatory, along with appropriate software for ordered shutdown.
2. Surge suppression on all phone lines, and also on serial/parallel lines that leave the room.
3. Line conditioners on any devices not connected to the UPS. If you do take a power hit, it's cheaper to replace a $50 line conditioner than a $1500 laser printer.

If this is too expensive for you, then downgrade the UPS/SPS to a line conditioner like the TrippLite. But don't go without at least that. Running unprotected is false economy, because you *will* lose equipment to electrical storms and, Murphy's Law being what it is, you will always get hit at the worst possible time.

An important question is "How do I know how big a UPS/SPS to get?" The watt rating of the UPS/SPS should be at least the sum of the peak ratings off all equipment connected to it (don't forget the console monitor). Power−supply marketroids tend to quote you capacities and formulas like "sum of VA ratings + 20%" which (surprise!) push you towards costlier hardware. Ignore them. If a watt rating is not given, watts = 0.75*VAmax.

One other consideration is that you typically shouldn't put a laser printer on a UPS toner heaters draw enough current to overload a UPS and cause a shutdown within seconds. The other thing is that you can't even put the laser printer on the same circuit with a UPS the heater kicks on every 20−30 seconds, and most UPSs will see the current draw as a brownout. So buy a separate line conditioner for the laser printer.

Finally, read the UPS's installation manual carefully if you're going to use it with other power−protection devices. Some UPSs don't like having surge suppressors between them and the equipment.

David personally recommends surge suppressors and line conditioners from Tripp−Lite (available both mail−order and retail), and UPSs from Best Power Technologies (Necedah, WI − 1−(800)−356−5737). I can enthusiastically second the TrippLite recommendation, but haven't dealt with Best Power at all.

Tripp−Lite has a whole range of products, from a $10 phone−line surge−suppressor, to line conditioners and SPSs with prces in the hundreds of dollars. They have a line of $50−80 line conditioners that are good for most peripherals (including your home stereo :−>).

Best Power Technologies sells two lines of UPSs in the range for home systems. The older and more expensive FERRUPS line (which is what David has) has a smart interface, and very good filtering and surge−suppression capabilities. He says "I have a 1.15kVA FERRUPS for my home system, which is overkill with my current hardware (although it rode out a 45 minute power failure with nary a whisper − no reboot). In 1990, I paid ~$1600 for this device, and that has since gone up. They also sell a newer line of Fortress UPSs. These are better suited in price for home systems. I don't know much about them, as they were not available when I bought my UPS. I expect that this is what most people will want to consider, though. In addition, Best sells Check−UPS, a software package (in source form) for monitoring the UPS and shutting it down. I have found Best to be a good company to deal with, with competent, knowledgeable sales people (who will be able to help you pick the right device), and helpful, courteous, and responsive technical support."

Other things to know:

A UPS should be wired directly to (or plugged directly into) the AC supply (i.e. a surge suppressor is neither required nor suggested between the wall and the UPS). In addition, a surge suppressor between the UPS and the equipment connected to it is redundant.

3.11. Radio Frequency Interference

(Thanks to Robert Corbett <Robert.Corbett@Eng.Sun.COM> for contributing much of this section)

Radio Frequency Interference (RFI) is a growing problem with PC−class machines. Today's processor speeds (above 33MHz) are such that the electromagnetic noise generated by a PC's circuitry in normal operation can degrade or jam radio and TV reception in the neighborhood. Such noise is called Radio Frequency Interference (RFI). Computers, as transmitting devices, are regulated by the Federal Communications Commission (FCC).

FCC regulations recognize two classes of computer:

If a PC is to be used in a home or apartment, it must be certified to be FCC class B. If it is not, neighbors have a legal right to prevent its use. FCC class A equipment is allowed in industrial environments.

Many systems are not FCC class B. Some manufacturers build boxes that are class B and then ship them with class A monitors or external disk drives. Even the cables can be a source of RFI.

It pays to be cautious. For example, the Mag MX17F is FCC class B. There are less expensive versions of the MX17 that are not. The Mag MX17 is a great monitor (I wish I had one). It would be painful to own one and not be allowed to use it.

An upgradeable system poses special problems. A system that is FCC class B with a 33 MHz CPU might not be when the CPU is upgraded to a 50 or 66 MHz CPU. Some upgrades require knockouts in the case to be removed. If a knockout is larger than whatever replaces it, RFI can leak out through the gap. Grounded metal shims can eliminate the leaks.

Even Class B systems don't mix will with wireless phonesets (not cellular phones, but the kind with a base station and antennaed headset). You'll often find a wireless phone hard to use withing 20 feet of a Class B machine.

To cut down on RFI, get a good metal case with tight joints, or at least make sure any plastic one you buy has a conductive lining. You can also strip the painted metal−to−metal contacting parts of paint so that there's good conductive metal contact. Paint's a poor conductor in most cases, so you can get some benefit from this.

4. Performance Tuning

Here are the places where you can trade off spending against the performance level you want to buy and your expected job mix.

4.1. How To Pick Your Processor

Right now, the chips to consider for running Unix are the Pentium IIs and Pentium IIIs and their clone equivalents from AMD or Cyrix. Life used to be more complicated, but with Pentium prices plunging as they have been and the PCI bus having taken over the world, nothing else makes much sense for a new Intel−based system.

Brands don't matter much, so don't feel you need to pay Intel's premiums if you see an attractive Cyrix, AMD or other chip−clone system offered.

To compare the performance of different Intel−based systems with each other and with machines from other manufacturers, you can take a look at the SPECmark Table at ftp://ftp.cdf.toronto.edu/pub/spectable. That document recommends (and I do too) that you read the SPEC FAQ at http://www.specbench.org/spec/specfaq.html to get background before browsing the table.

Good current advice about chipsets can be found at The Cheap /Linux/ Box.

4.2. Of Memory In...

Buy lots of RAM, it's the cheapest way to improve real performance on any virtual−memory system. 64MB now comes standard on most clone configurations. This is good enough for X.

Tuning is simple. Watch your job mix with top(1) and add memory until you're not swapping to disk any more.

4.3. Cache Flow

4.3.1. Overview

The most obscure of the important factors in the performance of a clone system is the motherboard's memory subsystem design, consisting both of primary and secondary cache and DRAM. The two questions performance−minded buyers have to deal with are: (1) does the cache design of a given motherboard work with Unix, and (2) how much cache SRAM should my system have, and how should it be organized?

Before normal clock speeds hit two digits in MHz, cache design wasn't a big issue. But DRAM's memory−cycle times just aren't fast enough to keep up with today's processors. Thus, your machine's memory controller caches memory references in faster static RAM (SRAM), reading from main memory in chunks that the board designer hopes will be large enough to keep the CPU continuously fed under a typical job load. If the cache system fails to work, the processor will be slowed down to less than the memory's real access speed   which, given typical 70ns DRAM parts, is about 7MHz.

You'll sometimes hear the terms L1, L2, and L3 cache. These refer to Level 1, Level 2, and Level 3 cache. L1, today, is always on the CPU (well, unless you're HP). L2 is off−chip cache. L3 is a second−level off−chip cache. Anything that will fit in L1 can be run at full CPU speed, as there is no need to go off chip. Anything (or things) too large to fit in L1 will try to run in L2. If it fits in L2, you still won't have to deal with the bus. If you're familiar with virtual memory, think of it this way: When you run out of L1, you swap to L2, when you run out of L2, you swap to main memory, when you run out of main memory, you swap to disk. Each stage is slower, and more prone to conflicts with other parts of the system, than what it follows.

Cache is like memory, but is faster, more expensive, and smaller. L1 is generally faster and smaller than L2, which is generally faster and smaller than L3, which is faster and smaller than memory. Some PCs will not have L2 or L3. Most workstation−class machines have L1 and L2. L3 is rare, even on big expensive Unix servers, but will become more common when CPUs start coming with L1 and L2 on−chip.

Because most L1 caches are on the CPU chip, there's isn't very much room for them so they tend to be small. It looks like the Pentium has 2 L1 caches, one for instructions (I−cache) and one for data (D−cache); each is 8 KB. If this is the only cache size available for Pentium, all laptops you look at will have this.

The size of the L2 cache you get will depend on what brand and model of laptop you buy, since Compaq and Fujitsu and NEC can decide independently how much L2 cache to put on their motherboards (within a range defined by the CPU chip). It's usually decided by the marketing people, not the technical people, based on what chips are available at what prices and what price they intend to sell the computer for. It looks like most benchmark results you'll see are with 256 or 512 KB of L2 cache; AT&T makes one Pentium−based server with 4 MB of L2 cache.

There are other cache−related buzzwords you may encounter.

"Write−back" means that when you update something in "memory" the cache doesn't actually push the new value out to the memory chips (or to L2, if it's an L1 write−back) until the "line" gets replaced in the cache. ("line" is the chunk−size caches act on, usually a small number of bytes like 8, 16, 32, or 64 for L1, or 32, 64, 128, or 256 for L2)

"Write−through" means that when you update "memory" the cache updates its value as well as sending an immediate update to physical memory (or to L2 if it's an L1 write−through). Write−back is generally faster if your application fits in the cache.

"Non−blocking, out of order" means that the CPU looks at the next N instructions it's about to execute. It executes the first and finds that the data isn't in cache. Since it's boring to just wait around for the data to come back from memory, it looks at the next instruction. If that 2nd instruction doesn't need the data the 1st instruction is waiting on, the CPU goes ahead and executes that instruction. If the 3rd instruction does need the data, it remembers it needs to execute that one after the data comes in and goes on to the 4th instruction. Depending on how many outstanding requests are allowed, if the 4th one causes a cache miss on a different line it may put that one on hold as well and go on to the 5th instruction. The Pentium Pro can do this, but I don't think the Pentium can.

"Set−associative" means the cache is split into 2 or more mini−caches. Because of the way things are accessed in a cache, this can help a program that has some "badly behaving" code mixed with some "good" code. Other terms that go with it are "LRU" (the mini−cache picked for replacement is the one Least Recently Used) or "random" (the line picked is selected randomly).

They can make a big difference in how happy you are with your system's performance. There are enough variables that you probably aren't going to be able to predict how happy you'll be with a configuration unless you sit down in front of the machine and run whatever it is you plan to run on it. Make up your own benchmark floppy with your primary application to take with you to showrooms. (Throw it away after all your test drives, since it will probably have collected a virus or three.)

Bigger or faster isn't always better. Speed is usually a tradeoff with size, and you have to match L2 cache size/speed to CPU speed. A system with a faster MHz CPU could perform worse than a system with a slower chip because the CPU<−−>L2 speed match might be such that the faster CPU requires a different, slower mode on the L2 connection.

If all you want to do is run MyLittleSpreadsheet, and the code and data all fit in 400 KB, a system with 512 KB of L2 cache will likely run more than twice as fast as a system with 256 KB of L2. If MLS fits in 600 KB and has a very sequential access pattern (a "cache−buster"), the 128 KB and 256 KB systems will perform about the same −− like a dog; if the pattern is random rather than sequential, the 512KB system will probably do some fractional amount better than the 128 KB system. This is why it's so important to try out your application and ignore impressive numbers for programs you're never going to run.

Also, you may find the Doom benchmark page useful :−).

4.3.2. How Caching Works

Caches get their leverage from exploiting two kinds of locality in memory references. Temporal locality means "access now, it should be accessed again soon", while spatial means "if byte N was asked for, byte N+1, N+2, N+3 will probably be wanted too". Because Unix multitasks, every context switch violates both types of locality. Spatial is impacted because contexts may not be located closely together. Temporal is impacted because you have to wait until all other ready contexts get their chance to run before you can run again.

One side−effect of what's today considered "good programming practice", with high−level languages using a lot of subroutine calls, is that the program counter of a typical process hops around like crazy. You might think that this, together with the periodic clock interrupts for multitasking, would make spatial locality very poor.

However, the clock interrupt only fires about 60 times per second. This is a very low overhead, if you consider how many instructions can be exectuted at 60 MHz in 1/60th of a second (for a poor estimate, something like 30 MIPS * 1/60 = half a million instructions−−at 16 bits each, roughly a megabyte of memory has been walked through!). This is lots of opportunity to take advantage of temporal locality −− and most programs are not so large that their time−critical parts won't fit inside a megabyte.

(Thanks to Michael T. Pins and Joan Eslinger for much of this section.)

4.3.3. A Short Primer on Cache Design

Before we go further in discussing specifics of the Intel processors we'll need some basic cache−design theory. (Some of this repeats and extends the Overview.)

Modern system designs have two levels of caching; a primary or internal cache right on the chip, and a secondary or external cache in high−speed memory (typically static RAM) off−chip. The internal cache feeds the processor directly; the external cache feeds the internal cache.

A cache is said to *hit* when the processor, or a higher−level cache, calls for a particular memory location and gets it. Otherwise, it *misses* and has to go to main memory (or at least the next lower level of cache) to fetch the contents of the location. A cache's *hit rate* is the percentage of time, considered as a moving average, that it hits.

The external cache is added to reduce the cost of an internal cache miss. To speed the whole process up, it must serve the internal cache faster than main memory would be able to do (to hide the slowness of main memory). Thus, we desire a very high hit rate in the secondary cache as well as very high bandwidth to the processor.

Obviously, secondary cache hit rate can be improved by making it bigger. It can also be increased by increasing the associativity factor (more on this later, but for now note that too much associativity can cost a big penalty).

A cache is divided up into *lines*. Typically, in an i486 system, each line is 4 to 16 bytes long (the i486 internal cache uses 16−byte lines; external line size varies). When the processor reads from an external−cache address that is not in the internal cache, that address and the surrounding 16 bytes are read into a line.

Each cache line has a *tag* associated with it. The tag stores the address in memory that the data in that cache line came from. (Plus a bit to indicate that this line contains valid data).

Some more important terms describing how caches interact with memory:

write−through

> it wouldn't do to let your cache get out of sync with main memory. The simplest way to handle this is to arrange that every write to cache generates the corresponding write to main store. In the simplest kind of "write−through" cache, then, you only get cache speedup on reads. But if the cache system includes write postings, writes will usually be fast too.

write posting

> Most write−through cache designs have a few `write posting' buffers that allow the cache to accept write data as fast as a write back cache for later writing to memory. So, as far as the processor is concerned, most writes will happen at zero wait states (the full cache speed), unless the processor issues enough writes in a short interval to cause the write posting buffers to fill up faster than they can be emptied.

write−back

> For each cache address range in DRAM, writes are done to cache only until a new block has to be fetched due to out−of−range access (at which point the old one is flushed to DRAM). This is much faster, because you get cache speedup on writes as well as reads. It's also more expensive and trickier to get right.

Write−back secondary caches are generally not a good idea. Beyond what was said in the write−through paragraph above, recall that the goal of the secondary cache is to have a high hit rate and high bandwidth to the processor's internal cache. When a cache−miss occurs in the secondary cache, often the line being replaced is dirty and must be written to main memory first. The total time to service the secondary cache miss nearly doubles.

Even when the secondary cache line being replaced is not dirty, the service time goes up because the dirty bit must first be examined before accessing to main memory. Write−through caches have the advantage of being able to look up data in the secondary cache and in main memory in parallel (in the case where the secondary cache misses, some of the delay of looking in main memory has already been taken away). (Write−back caches cannot do this because they might have to write−back the cache line before doing the main memory read.)

For these reasons, write−back caches are generally regarded as being inferior to write−posting buffers. They cost too much silicon and more often than not perform worse.

Now some terms that describe cache organization. To understand these, you need to think of main RAM as being divided into consecutive, non−overlapping segments we'll call "regions". A typical region size is 64K. Each region is mapped to a cache line, 4 to 128 bytes in size (a typical size is 16). When the processor reads from an address in a given region, and that address is not already in core, the location and others near it are read into a line.

direct−mapped

> Describes a cache system in which each region has exactly one corresponding line (also called "one−way cache").

two−way set−associative

> Each region has *two* possible slots; thus your odds of not having to fetch from DRAM (and, hence, your effective speed) go up.

There are also "four−way" caches. In general, an n−way cache has n pages per region and improves your effective speed by some factor proportional to n. However, multiset caches become very costly in terms of silicon real estate, so one does not commonly see five−way or higher caches.

Because set−associative caches make better use of SRAM, they typically require less SRAM than a direct−mapped cache for equivalent performance. They're also less vulnerable to Unix's heavy memory usage. Andy Glew of USENET's comp.arch group says "the usual rule of thumb is that a 4−way set−associative cache is equivalent to a direct−mapped cache of twice the size". On the other hand, some claim that as cache size gets larger, two−way associativity becomes less useful. According to this school of thought it actually becomes a net loss over a direct−mapped cache at cahe sizes over 256K.

So, typically, you see multi−set cache designs on internal caches, but direct−mapped designs for external caches.

The larger you make a cache line, the cheaper the design will be, as you save on expensive tag ram, but the worse the performance will be, as you pay for each cache miss manyfold. It is not reasonable to have 32 bytes reload on a cache miss.

In the presence of interleaved DRAM memory, a cache line should not be larger than a whole DRAM line −− double interleaved: 2*4 bytes, quadruple 4*4 bytes. Otherwise, memory fetches to the external cache get slow.

An external cache which can support the i486 burst mode can increase bandwidth to a much higher level than one which doesn't, and can significantly reduce the cost of an internal cache miss.

4.4. Suggestions for Buying

The best advice your humble editor can give is a collection of rules of thumb. Your mileage may vary...

4.4.1. Rule 1: Buy only motherboards that have been tested with Unix

One of DOS's many sins is that it licenses poor hardware design; it's too brain−dead to stretch the cache system much. Thus, bad cache designs that will run DOS can completely hose Unix, slowing the machine to a crawl or even (in extreme cases) causing frequent random panics. Make sure your motherboard or system has been tested with some Unix variant.

4.4.2. Rule 2: Be sure you get enough cache.

If your motherboard offers multiple cache sizes, make sure you how much is required to service the DRAM you plan to install.

Bela Lubkin writes: "Excess RAM [over what your cache can support] is a very bad idea: most designs prevent memory outside the external cache's cachable range from being cached by the 486 internal cache either. Code running from this memory runs up to 11 times slower than code running out of fully cached memory."

4.4.3. Rule 3: "Enough cache" is at least 64K per 16MB of DRAM

Hardware caches are usually designed to achieve effective 0 wait state status, rather than perform any significant buffering of data. As a general rule, 64Kb cache handles up to 16Mb memory; more is redundant.

A more sophisticated way of determining cache size is to estimate the number of processes you expect to be running simultaneously (ie 1 + expected load average, let this value be N). Your external cache should be about N * 32k in size. The justification for this is as follows: upon a context switch, it is a good idea to be able to hold the entire i486 internal cache in the secondary cache. For each process you would need something less than 8k * 4 (since it is 4−set associative, you need 32k to help map the conflicting cache lines, and the extra cache left over (24k) should be plenty to help improve the hit rate of the secondary cache when the internal cache misses. The number of main memory accesses caused by context switching should be reduced.

Of course, if you are going to be running programs with large memory requirements (especially data), then a huge secondary cache would probably be a big win. But most programs in the run queue will be small though (ls, cat, more, etc.).

4.4.4. Rule 4: If possible, max out the board's cache when you first buy

Bela continues: "Get the largest cache size your motherboard supports, even if you're not fully populating it with RAM. The motherboard manufacturer buys cache chips in quantity, knows how to install them correctly, and you won't end up throwing out the small chips later when you upgrade your main RAM."

Gerard Lemieux qualifies this by observing that if adding SRAM increases the external cache line size, rather than increasing the number of cache lines, it's a lose. If this is the case, then an external cache miss could cost you dearly; imagine how long the processor would have to wait if the line size grew to 1024 bytes. If the cache has a poor hit rate (likely true, since the number of lines has not changed), performance would deteriorate.

Also (he observes), spending an additional $250 for cache chips might buy you 2−3% in performance (even in Unix). You must ask yourself if it is really worth it.

4.4.5. Caveat

A lot of fast chips are held back by poor cache systems and slow memory. To avoid trouble, cloners often insert wait states at the cache, slowing down the chip to the effective speed of a much slower chip.

(Thanks to Guy Gerard Lemieux <lemieux@eecg.toronto.edu> for insightful comments on an earlier version of this section.)

4.5. Bus Wars

This is yet another area in which progress has simplified your choices a lot. There used to be no fewer than four competing bus standards out there (ISA, EISA, VESA/VLB, PCI, and PCMCIA). Now there are effectively just two −− PCI for desktop/tower machines and PCMCIA for laptops.

4.5.1. Bus Types

PCI is Intel's fast 64−bit bus for the Pentium. Many PCI boards are actually PCI/ISA that supports both standards, so you can use less expensive ISA peripherals and controllers. Beware, though; dual−bus boards lose about 10% of their performance relative to single−bus PCI boards.

In the laptop market everything is PCMCIA. PCMCIA peripherals are about the size of credit cards (85x54mm) and vary in thickness between 5 and 10mm. They have the interesting feature that they can be hot−swapped (unplugged out and plugged in) while the computer is on. However, they are seldom seen in desktop machines. They require a special daemon to handle swapping; free versions are now standard under Linux.

4.5.2. Plug And Play

Many PCI cards have a feature called ``Plug and Play''. These cards negotiate with the operating system at boot time for things like IRQs and DMA channels −− they have no jumpers. Beware of these! Linux doesn't yet have full support for Plug and Play, though there are support utilities available. (Of the Microsoft OSs, only Windows 95 and up supports Plug and Play fully −− DOS can't handle it at all and Windows 3.1 requires manual intervention).

4.5.3. Historical Note

There used to be two ISA buses, the original 8−bit IBM PC and a 16−bit compatible extension sometimes called "AT bus". The term ISA didn't come into use until well into the lifetime of the latter. Here's a more complete list:

ISA

The original IBM PC bus architecture. The 8−bit version is completely extinct. The 16−bit AT version is still alive but has been declared obsolete by Intel in the PC99. specification.

MCA

Micro−Channel Architecture. A ``standard'' that IBM attempted to promulgate, esp. in the PS/2 series of machines. While they tried to claim it was faster/more efficient, it really was only marginally better than ISA. Its real advantage−−to IBM−−was that it was a closed architecture; they didn't publish the details of implementation as they had on virtually everything for the XT/AT. It failed horribly; customers didn't want to walk back into that trap.

EISA

Extended ISA. Required motherboard setup and manufacturer−provided descriptions that got loaded into flash ROM. Manually. By the user. Superseded by VLB years ago and now extinct.

VLB

VESA Local Bus. Usually seen on video cards providing high−speed graphics data transfer, it was also being touted for other cards. VLB slots accepted ISA and EISA cards and a further extension called the VESA local−bus specification. Supplanted by PCI.

PCI

Peripheral Component Interconnect. The winner of the bus wars.

4.6. Disk Wars: IDE vs. SCSI

4.6.1. Overview

Another basic decision is IDE vs. SCSI. Either kind of disk costs about the same, but the premium for a SCSI card varies all over the lot, partly because of price differences between VLB and PCI SCSI cards and especially because many motherboard vendors bundle an IDE chipset right on the system board. SCSI gives you better speed and throughput and loads the processor less, a win for larger disks and an especially significant consideration in a multi−user environment; also it's more expandable.

In terms of pure disk speed, IDE will always be faster, as they use the same underlying disks, and IDE has less overhead. As fast as disks are getting today, the difference is effectively noise. The real advantage of SCSI comes from its extra brains. IDE uses polled I/O, which means that when you are accessing the disk, the CPU isn't doing anything else. Most SCSI systems, on the other hand, are DMA based, freeing up the system to do other things at the same time. Hence, in terms of full system performance, SCSI is indeed faster if you have good hardware and an intelligent OS.

Another important win for SCSI is that it handles multiple devices much more efficiently. You can have at most two IDE devices; four for EIDE. SCSI permits up to 7 (15 for Wide SCSI).

If you have two IDE (or ST506 or ESDI) drives, only one can transfer between memory and disk at once. In fact, you have to program them at such a low level that one drive might actually be blocked from *seeking* while you're talking to the other drive. SCSI drives are mostly autonomous and can do everything at once; and current SCSI drives are not quite fast enough to flood more than half the SCSI bus bandwidth, so you can have at least two drives on a single bus pumping full speed without using it up. In reality, you don't keep drives running full speed all the time, so you should be able to have 3−4 drives on a bus before you really start feeling bandwidth crunch.

Of course, IDE is cheaper. Many motherboards have IDE right on board now; if not, you'll pay maybe $15 for an IDE adapter board, as opposed to $200+ for the leading SCSI controller. Also, the cheap SCSI cabling most vendors ship can be flaky. You have to use expensive high−class cables for consistently good results. See Mark Sutton's horror story.

4.6.2. Enhanced IDE

These days you seldom see plain IDE; souped−up variants are more usual. These are "Enhanced IDE" (E−IDE) and "Fast AT Attachment" (usually ATA for short). ATA is Seagate's subset of E−IDE, excluding some features designed to permit chaining with CD−ROMs and tape drives using the new "ATAPI" interface (an E−IDE extension; so far only the CD−ROMs exist); in practice, ATA and E−IDE are identical.

You'll need to be careful about chaining in CD−ROMs and tape drives when using IDE/ATA. The IDE bus sends all commands to all disks; they're supposed to latch, and each drive then checks to see whether it is the intended target. The problem is that badly−written drivers for CD−ROMs and tapes can collide with the disk command set. It takes expertise to match these peripherals.

Neither ATA nor E−IDE has the sustained throughput capacity of SCSI (they're not designed to) but they are 60−90% faster than plain old IDE. E−IDE's new ``mode 3'' boosts the IDE transfer rate from IDE's 3.3MB/sec to 13.3MB/sec. The new interface supports up to 4 drives of up to 8.4 gigabytes capacity.

E−IDE and ATA are advertised as being completely compatible with old IDE. Theoretically, you can mix IDE, E−IDE and ATA drives and controllers any way you like, and the worst result you'll get is conventional IDE performance if the enhancements don't match up (the controller picks the lowest latch speed). In practice, some IDE controllers (notably the BusLogic) choke on enhanced IDE.

Accordingly, I recommend against trying to mix device types an an E−IDE/ATA bus. Unfortunately, this removes much of E−IDE/ATA's usefulness!

E−IDE on drives above 540MB does automatic block mapping to fool the BIOS about the drive geometry (avoiding limits in the BIOS type tables). They don't require special Unix drivers.

Many motherboards now support ``dual EIDE'' channels, i.e. two separate [E]IDE interfaces each of which can, theoretically, support two IDE disks or ATA−style devices.

4.6.3. SCSI Terminology

The following, by Ashok Singhal <ashoks@duckjibe.eng.sun.com> of Sun Microsystems with additions by your humble editor, is a valiant attempt to demystify SCSI terminology.

The terms ``SCSI'', ``SCSI−2'', and ``SCSI−3'' refer to three different specifications. Each specification has a number of options. Many of these options are independent of each other. I like to think of the main options (there are others that I'll skip over because I don't know enough about them to talk about them on the net) by classifying them into five categories:

4.6.3.1. Logical: SCSI−1, SCSI−2, SCSI−3>

This refers to the commands that the controllers understand. Shortly after SCSI first came out, the vendors agreed on a spec for a common comand set called CCS. CCS was made a required part of the SCSI−2 standard. You should be able to use a SCSI disk with a SCSI−2 card and vice−versa as long as they both support CCS. Non−CCS SCSI devices aren't worth considering.

``SCSI−3'' is a superset of SCSI−2 including commands intended for CD−R and streaming multimedia devices.

4.6.3.2. Electrical Interface

- single−ended (max cable length 6 meters)
- differential (max cable length 25 meters)

This option is independent of command set, speed, and path width. Differential is less common but allows better noise immunity and longer cables. It's rare in SCSI−1 controllers.

For a PC you will probably always see single−ended SCSI controllers but if you're shopping around for disks you might run across differential disks. They will likely be more expensive than single−ended ones and will not work on your single−ended bus.

4.6.3.3. Handshake

- Asynchronous (acknowledge each word (8, 16 or 32 bits) transferred.
- Synchronous (multiple−word transfers permitted between ACKS).

Synchronous is faster. This mode is negotiated between controller and device; modes may be mixed on the same bus. This is independent of command set, data width, and electrical interface.

4.6.3.4. Synchronous Speed (does not apply for asynchronous option)

Normal transfer speed is 5 megabytes/sec. The ``fast'' option (10 mb/sec) is defined only in SCSI−2 and SCSI−3. Fast−20 (or ``Ultra'') is 20 mb/sec; Fast−40 (or "Ultra−2") is 40MB/sec. The fast options basically defines shorter timing parameters such as the assertion period and hold time.

The parameters of the synchronous transfer are negotiated between each target and initiator so different speed transfers can occur over the same bus.

4.6.3.5. Path width

The standard SCSI data path is 8 bits wide. The ``wide'' option exploits a 16− or 32−bit data path (uses 68−pin rather than 50−pin data cables). You also get 4−bit rather than 3−bit device IDs, so you can have up to 16 devices. The wide option doubles or quadruples your transfer rate, so for example a fast−20/wide SCSI link using 16 bits transfers 40mb/sec.

What are those ``LUN'' numbers you see when you boot up? Think of them as sub−addresses on the SCSI bus. Most SCSI devices have only one ``logical'' device inside them, thus they're LUN zero. Some SCSI devices can, however, present more than one separate logical unit to the bus master, with different LUNs (0 through 7). The only context in which you'll normally use LUNs is with CD−ROM juke boxes. Some have been marketed that offer up to 7 CD−ROMS with one read head. These use the LUN to differentiate which disk to select.

(There's history behind this. Back in the days of EISA, drives were supposed to be under the control of a separate SCSI controller, which could handle up to 7 such devices (15 for wide SCSI). These drives were to be the Logical Units; hence the LUN, or Logical Unit Number. Then, up to 7 of these SCSI controllers would be run by the controller that we today consider the SCSI controller. In practice, hardware cost dropped so rapidly, and capability increased so rapidly, it became more logical to embed the controller on the drive.)

4.6.4. Avoiding Pitfalls

Here are a couple of rules and heuristics to follow:

Rule 1: Total SCSI cable length (both external and internal devices) must not exceed six meters. For modern Ultra SCSI (with its higher speed) cut that to three feet!

It's probably not a good idea to cable 20MB/s or faster SCSI devices externally at all. If you must, one of our informants advises using a Granite Digital ``perfect impedance'' teflon cable (or equivalent); these cables basically provide a near−perfect electrical environment for a decent price, and can be ordered in custom configurations if needed.

A common error is to forget the length of the ribbon cable used for internal devices when adding external ones (that is, devices chained to the SCSI board's external connector).

Rule 2: Both ends of the bus have to be electrically terminated.

On older devices this is done with removable resistor packs   typically 8−pin−inline widgets, yellow or blue, that are plugged into a plastic connector somewhere near the edge of the PCB board on your device. Peripherals commonly come with resistor packs plugged in; you must *remove* the packs on all devices except the two end ones in the physical chain.

Newer devices advertised as having "internal termination" have a jumper or switch on the PCB board that enables termination. These devices are preferable, because the resistor packs are easy to lose or damage.

Rule 3: No more than seven devices per chain (fifteen for Wide SCSI).

There are eight SCSI IDs per controller. The controller reserves ID 7 or 15, so your devices can use IDs 0 through 6 (or 0 through 14, wide). No two devices can share an ID; if this happens by accident, neither will work.

The conventional ID assignments are: Primary hard disk = ID 0, Secondary hard disk = ID 1, Tape = ID 2. Some Unixes (notably SCO) have these wired in. You select a device's ID with jumpers on the PCB or a thumbwheel.

SCSI IDs are completely independent of physical device chain position.

Heuristic 1: Stick with controllers and devices that use the Centronics−style 50−pin connector. Internally these connectors are physically identical to diskette cables. Externally they use a D50 shell. This "standard" connector is common in the desktop/tower/rackmount−PC world, but you'll find lots of funky DIN and mini−DIN plugs on devices designed for Macintosh boxes and some laptops. Ask in advance and don't get burned.

Heuristic 2: For now, when buying a controller, go with an Adaptec xx42 or one of its clones such as the BusLogic 542. (I like the BusLogic 946 and 956, two particularly fast Adaptec clones well−supported under Linux.) The Adaptec is the card everybody supports and the de−facto standard. Occasional integration problems have been reported with Unix under Future Domain and UltraStor cards, apparently due to command−set incompatibilities. At least, before you buy these, make sure your OS explicitly supports them.

However: Beware the combination of an Adaptec 1542 with a PCI Mach32 video card. Older (1.1) Linux kernels handled it OK, but all current ones choke. Your editor had to replace his 1542 because of this, swearing sulphurously the while.

Heuristic 3: You'll have fewer hassles if all your cables are made by the same outfit. (This is due to impedence reflections from minor mismatches. You can get situations where cable A will work with B, cable B will work with C, but A and C aren't happy together. It's also non−commutative. The fact that `computer to A to B' works doesn't mean that `computer to B to A' will work.

Heuristic 4. Beware Cheap SCSI Cables!

Mark Sutton tells the following instructive horror story in a note dated 5 Apr 1997:

I recently added an additional SCSI hard drive to my home machine. I bought an OEM packaged Quantum Fireball 2 gig SCSI drive (meaning, I bought a drive in shrinkwrap, without so much as mounting hardware or a manual. Thank God for Quantum's web page or I would have had no idea how to disable termination or set the SCSI ID on this sucker. Anyway, I digress...). I stuck the drive in an external mounting kit that I found in a pile of discarded computer parts at work and my that boss said I could have. (All 5 of my internal bays were full of devices.)

Anyway, I had my drive, and my external SCSI mounting kit, I needed a cable.

I went into my friendly local CompUSA in search of a SCSI cable, and side−by−side, on two hooks, were two "identical" SCSI cables. Both were 3 feet. Both had centronics to centronics connectors, both were made by the same manufacturer. They had slightly different model numbers. One was $16.00, one was $30.00. Of course, I bought the $16 cable.

Bad, I say, BAD BAD MISTAKE. I hooked this sucker up like so:

```
   –            –
|Internal|––|Adaptec| |New Quantum| |UMAX    |
|Devices |  |1542CF | ^ |  Disk     | ^ |Scanner|
   –    ––  |     – |
                      |               |
                  New $16 cable   Cable that came
                                    with scanner.
```

Shortly after booting, I found that data all over my old internal hard drive was being hosed. This was happening in DOS as well as in Linux. Any disk access on either disk was hosing data on both disks, attempts to scan were resulting in corrupted scans *and* hosing files on the hard disks. By the time I finished swapping cables around, and checking terminations and settings, I had to restore both Linux and DOS from backups.

I went back to CompUSA, exchanged the $16 cable for the $30 one, hooked it up and had no more problems.

I carefully examined the cables and discovered that the $30 cable contained 24 individual twisted pairs. Each data line was twisted with a ground line. The $16 cable was 24 data wires with one overall grounded shield. Yet, both of these cables (from the same *manufacturer*) were being sold as SCSI cables!

You get what you pay for.

(Another correspondent guesses that the cheap cable probably said ``Macintosh" on it. The Mac connector is missing most of its ground pins.)

4.6.5. Trends to Watch For

Disks of less that 2GB capacity simply aren't being manufactured anymore; there's no margin in them. Our spies tell us that all major disk makers retooled their lines a while back to produce 540MB unit platters, which are simply being stacked 2N per spindle to produce ranges of drives with roughly 1GB increments of capacity. The highest reasonably−priced drives are still 9GB (16 platters per drive), but you can get 23GB or even 45GB capacities (these are probably packing 2.4GB per platter).

Average drive latency is inversely proportional to the disk's rotational speed. For years, most disks spun at 3600 rpm; most high−performance disks now spin at 7,200 rpm, and high−end disks like the Seagate Cheetah line are moving to 10,000 rpm. These fast−spin disks run extremely hot; expect cooling to become a critical constraint in drive design.

Drive densities have reached the point at which standard inductive read/write heads are a bottleneck. In newer designs, expect to see magnetoresistive head assemblies with separate read and write elements.

4.6.6. More Resources

There's a USENET SCSI FAQ. Also see the home page of the T10 committee that writes SCSI standards.

There is a large searchable database of hard disk and controller information at the PC DISK Hardware Database.

4.7. Other Disk Decisions

Look at seek times and transfer rates for your disk; under Unix disk speed and throughput are so important that a 1−millisecond difference in average seek time can be noticeable.

4.7.1. Disk Brands

An industry insider (a man who buys hard drives for systems integration) has passed us some interesting tips about drive brands. He says the absolute best−quality drives are the Hewlett−Packards but you will pay a hefty premium for that quality.

The other top−tier manufacturers are Quantum and Seagate; these drives combine cutting−edge technology with very aggressive pricing.

The second tier consists of Maxtor, Conner, and Western Digital.

Maxtor often leads in capacity and speed, but at some cost in other quality measures. For example, many of the high−capacity Maxtor drives have serious RFI emission problems which can cause high error rates. SCSI has built−in ECC correction, so SCSI drives only take a performance hit from this; but it can lead to actual errors from IDE drives.

Western Digital sells most of its output to Gateway at sweetheart prices; WD drives are thus not widely available elsewhere.

The third tier consists of Fujitsu, Toshiba, and everyone else. My friend observes that the Japanese, despite their reputation for process engineering savvy, are notably poor at drive manufacturing; they've never spent the money and engineering time needed to get really good at making the media.

If you see JTS drives on offer, run away. It is reliably reported that they are horrible.

Just as a matter of interest, he also says that hard drives typically start their life cycle at an OEM price around $400 each. When the price erodes to around $180, the product gets turfed   there's no margin any more.

I've found a good cheap source for reconditioned SCSI disks at Uptime Computer Support Services.

4.7.2. To Cache Or Not To Cache?

Previous issues said "Disk cacheing is good, but there can be too much of a good thing. Excessively large caches will slow the system because the overhead for cache fills swamps the real accesses (this is especially a trap for databases and other applications that do non−sequential I/O). More than 100K of cache is probably a bad idea for a general−purpose Unix box; watch out for manufacturers who inflate cache size because memory is cheap and they think customers will be impressed by big numbers." This may no longer be true on current hardware; in particular, most controllers will interrupt a cache−fill to fulfill a `real' read request.

In any case, having a large cached hard drive (particularly in the IDEs) often does not translate to better performance. For example, Quantum makes a 210Mb IDE drive which comes with 256Kb cache. Conner and Maxtor also have 210Mb drives, but only with 64Kb caches. The transfer rate on the drives, however, show that the Quantum comes in at 890Kb/sec, while the Maxtor and Conner fly away at 1200Kb/sec. Clearly, the Conner and Maxtor make much better use of their smaller caches.

However, it may be that *any* hardware disk cacheing is a lose for Unix! Scott Bennett <bennett@mp.cs.niu.edu> reports a discussion on comp.unix.wizards: "nobody found the hardware disk caches to be as effective in terms of performance as the file system buffer cache...In many cases, disabling the hardware cache improved system performance substantially. The interpretation of these results was that the cacheing algorithm in the kernel was superior to, or at least better tuned to Unix accesses than, the hardware cacheing algorithms."

On the other hand, Stuart Lynne <sl@mimsey.com> writes:

Ok. What I did was to use the iozone program.

What this showed was that on my root disk in single user mode I could get about 500kb for writing and 1000kb for reading a 10MB file. With the disk cache disabled I was able to get the same for writing but only about 500kb for reading. I.e. it appears the cache is a win for reading, at least if you have nothing else happening.

Next I used a script which started up iozone in parallel on all four disks, two to each of the big disks (three) and one on the smaller disk. A total of seven iozone's competing with each other.

This showed several interesting results. First it was apparant that higher numbered drives *did* get priority on the SCSI bus. They consistantly got better throughput when competing against lower numbered drives. Specifically drive 1 got better results than drive 0 on controller 0. Drive 4 got better results than drive 3 on controller 1. All of the drives are high end Seagate and have similiar characteristics.

In general with cache enabled the results where better for reading than writing. When the cache was disabled the write speed in some cases went up a bit and the read speed dropped. It would seem that the readahead in some cases can compete with the writes and slow them down.

My conclusions are that we'll see better performance with the cache. First the tendency is to do more reading than writing in your average Unix system so we probably want to optimize that. Second if we assume an adequate system cache slow writes shouldn't affect an individual process much. When we write we are filling the cache and we don't usually care how long it takes to get flushed. Of course we would notice it when writing very large files.

Thus (this is your humble editor again), I can only recommend experiment. Try disabling the cache. Your throughput may go up!

4.8. Tuning Your I/O Subsystem

(This section comes to us courtesy of Perry The Cynic, <perry@sutr.cynic.org>. My own experience agrees pretty completely with his.)

Building a good I/O subsystem boils down to two major points: *pick matched components* so you don't over−build any piece without benefit, and *construct the whole pipe such that it can feed what your OS/application combo needs*.

It's important to recognize that ``balance'' is with respect to not only a particular processor/memory subsystem, but also to a particular OS and application mix. A Unix server machine running the whole TCP/IP server suite has radically different I/O requirements than a video−editing workstation. For the ``big boys'' a good consultant will sample the I/O mix (by reading existing system performance logs or taking new measurements) and figure out how big the I/O system needs to be to satisfy that app mix. This is not something your typical Linux buyer will want to do; for one, the application mix is not static and will change over time. So what you'll do instead is design an I/O subsystem that is internally matched and provides maximum potential I/O performance for the money you're willing to spend. Then you look at the price points and compare them with those for the memory subsystem. That's the most important trade−off inside the box.

So the job now is to design and buy an I/O subsystem that is well matched to provide the best bang for your buck. The two major performance numbers for disk I/O are latency and bandwidth. Latency is how long a program has to wait to get a little piece of random data it asked for. Bandwidth is how much contiguous data can be sent to/from the disk once you've done the first piece. Latency is measured in milliseconds (ms); bandwidth in megabytes per second (MB/s). Obviously, a third number of interest is how big all of your disks are together (how much storage you've got), in Gigabytes (GB).

Within a rather big envelope, minimizing latency is the cat's meow. Every millisecond you shave off effective latency will make your system feel significantly faster. Bandwidth, on the other hand, only helps you if you suck a big chunk of contiguous data off the disk, which happens rarely to most programs. You have to keep bandwidth in mind to avoid mis−matching pieces, because (obviously) the lowest usable bandwidth in a pipe constrains everything else.

I'm going to ignore IDE. IDE is no good for multi−processing systems, period. You may use an IDE CD−ROM if you don't care about its performance, but if you care about your I/O performance, go SCSI.

Let's look at the disks first. Whenever you seriously look at a disk, *get its data sheet*. Every reputable manufacturer has them on their website; just read off the product code and follow the bouncing lights. Beware of numbers (`<12ms fast!') you may see in ads; these folks often look for the lowest/highest numbers on the data sheet and stick them into the ad copy. Not dishonest (usually), but ignorant.

What you need to find out for a disk is:

1. What kind of SCSI interface does it have? Look for "fast", "ultra", and "wide". Ignore disks that say "fiber" or "differential" (these are specialty physical layers not appropriate for the insides of small computers). Note that you'll often find the same disk with different interfaces.
2. What is the "typical seek" time (ms)? Make sure you get "typical", not "track−to−track" or "maximum" or some other measure (these don't relate in obvious ways, due to things like head−settling time).
3. What is the rotational speed? This is typically 4500, 5400, 7200, or 10000 rpm (rotations per minute). Also look for "rotational latency" (in ms). (In a pinch, average rotational latency is approx. 30000/rpm in milliseconds.)
4. What is the `media transfer rate' or speed (in MB/s)? Many disks will have a range of numbers (say, 7.2−10.8MB/s). Don't confuse this with the "interface transfer rate" which is always a round number (10 or 20 or 40MB/s) and is the speed of the SCSI bus itself.

These numbers will let you do apple−with−apples comparisons of disks. Beware that they will differ on different−size models of the same disk; typically, bigger disks have slower seek times.

Now what does it all mean? Bandwidth first: the `media transfer rate' is how much data you can, under ideal conditions, get off the disk per second. This is a function mostly of rotation speed; the faster the disk rotates, the more data passes under the heads per time unit. This constrains the sustained bandwidth of *this disk*.

More interestingly, your effective latency is the sum of typical seek time and rotational latency. So for a disk with 8.5ms seek time and 4ms rotational latency, you can expect to spend about 12.5ms between the moment the disk `wants' to read your data and the moment when it actually starts reading it. This is the one number you are trying to make small. Thus, you're looking for a disk with low seek times and high rotation (RPM) rates.

For comparison purposes, the first hard drive I ever bought was a 20MB drive with 65ms seek time and about 3000RPM rotation. A floppy drive has about 100−200ms seek time. A CD−ROM drive can be anywhere between 120ms (fast) and 400ms (slow). The best IDE harddrives have about 10−12ms and 5400 rpm. The best SCSI harddrive I know (the Seagate Cheetah) runs 7.8ms/10000rpm.

Fast, big drives are expensive. Really big drives are very expensive (that's 20GB+ drives as of this writing in August 1998). Really fast drives are pretty expensive (that's about < 8ms right now). On the other end, really slow, small drives are cheap but not cost effective, because it doesn't cost any less to make the cases, ship the drives, and sell them.

In between is a `sweet spot' where moving in either direction (cheaper or more expensive) will cost you more than you get out of it. The sweet spot moves (towards better value) with time. Right now (August 1998), it's about at 4GB drives, 8−10ms, 5400−7200rpm, fast or ultra SCSI. If you can make the effort, go to your local computer superstore and write down a dozen or so drives they sell `naked'. (If they don't sell at least a dozen hard drives naked, find yourself a better store. Use the Web, Luke!) Plot cost against size, seek and rotational speed, and it will usually become pretty obvious which ones to get for your budget.

Do look for specials in stores; many superstores buy overstock from manufacturers. If this is near the `sweet spot', it's often surprisingly cheaper than comparable drives. Just make sure you understand the warranty procedures.

Note that if you need a lot of capacity, you may be better off with two (or more) drives than a single, bigger one. Not only can it be cheaper (2x4GB is often cheaper than 1x9GB), but you end up with two separate head assemblies that move independently, which can cut down on latency quite a bit (see below).

Once you've decided which kind of drive(s) you want, you must decide how to distribute them over one or more SCSI buses. Yes, you *may* want more than one SCSI bus. (My current desktop machine has three.) Essentially, the trick is to make sure that all the disks on one bus, talking at the same time, don't exceed the capacity of that bus. At this time, I can't recommend anything but an Ultra/Wide SCSI controller. This means that the attached SCSI bus can transfer data at up to 40MB/s for an Ultra/Wide disk, 20MB/s for an Ultra/narrow disk, and 10MB/s for a `fast SCSI' disk. These numbers allow you do do your math: an 8MB/s disk will eat an entire bus on its own if it's `fast' (10MB/s). Three 6MB/s ultra/narrow disks fit onto one bus (3x6=18MB/s<20MB/s), but just barely. Two ultra/wide Cheetahs (12.8MB/s) will share an (ultra/wide) bus (25.6<40), but they would collide on an ultra/narrow bus, and any one Cheetah would be bandwidth constrained on a (non−ultra) `fast' bus (12.8 > 10).

If you find that you need two SCSI buses, you can go with `dual channel' versions of many popular SCSI controller cards (including the Adaptec). These are simply two controllers on one card (thus taking only one

PCI slot). This is cheaper and more compact than two cards; however, on some motherboards with more than 3 PCI slots, using two cards may be somewhat faster (ask me what a PCI bridge is :−).

How do you deal with slow SCSI devices − CD−ROMS, scanners, tape drives, etc.? If you stick these onto a SCSI bus with fast disks, they will slow down things a bit. You can either accept that (as in ``I hardly ever use my scanner anyway''), or stick them onto a separate SCSI bus off a cheap controller card. Or you can (try to) get an ATA version to stick onto that inevitable IDE interface on your motherboard. The same logic applies to disks you won't normally use, such as removables for data exchange.

If you find yourself at the high end of the bandwidth game, be aware that the theoretical maximum of the PCI bus itself is 132MB/s. That means that a dual ultra/wide SCSI controller (2x40MB/s) can fill more than half of the PCI bus's bandwidth, and it is not advised to add another fast controller to that mix. As it is, your device driver better be well written, or your entire system will melt down (figuratively speaking).

Incidentally, all of the numbers I used are `optimal' bandwidth numbers. The real scoop is usually somewhere between 50−70% of nominal, but things tend to cancel out − the drives don't quite transfer as fast as they might, but the SCSI bus has overhead too, as does the controller card.

Whether you have a single disk or multiple ones, on one or several SCSI buses, you should give careful thought to their partition layout. Given a set of disks and controllers, this is the most crucial performance decision you'll make.

A partition is a contiguous group of sectors on the disk. Partitioning typically starts at the outside and proceeds inwards. All partitions on one disk share a single head assembly. That means that if you try to overlap I/O on the first and last partition of a disk, the heads must move full stroke back and forth over the disk, which can radically increase seek time delays. A partition that is in the middle of a partition stack is likely to have best seek performance, since at worst the heads only have to move half−way to get there (and they're likely to be around the area anyway).

Whenever possible, split partitions that compete onto different disks. For example, /usr and the swap should be on different disks if at all possible (unless you have outrageous amounts of RAM).

Another wrinkle is that most modern disks use `zone sectoring'. The upshot is that outside partitions will have higher bandwidth than inner ones (there is more data under the heads per revolution). So if you need a work area for data streaming (say, a CD−R pre−image to record), it should go on an outside (early numbered) partition of a fast−rotating disk. Conversely, it's a good convention to put rarely−used, performance−uncritical partitions on the inside (last).

Another notes concerns SCSI mode pages. Each (modern) SCSI disk has a small part of its disk (or a dedicated EEPROM) reserved for persistent configuration information. These parameters are called `mode pages', for the mechanism (in the SCSI protocol) for accessing them. Mode page parameters determine, among others, how the disk will write−cache, what forms of error recovery it uses, how its RAM cache is organized, etc. Very few configuration utilities allow access to mode page parameters (I use FWB Toolkit on a Mac − it's simply the best tool I know for that task), and the settings are usually factory preset for, uh, Windows 95 environments with marginal hardware and single−user operation. Particularly the cache organization and disconnect/reconnect pages can make a tremendous difference in actual performance. Unfortunately there's really no easy lunch here − if you set mode page parameters wrong, you can screw up your data in ways you won't notice until months later, so this is definitely `no playing with the pushebuttons' territory.

Ah yes, caches. There are three major points where you could cache I/O buffers: the OS, the SCSI controller, and the on−disk controller. Intelligent OS caching is by far the biggest win, for many reasons. RAM caches on SCSI controller cards are pretty pointless these days; you shouldn't pay extra for them, and experiment with disabling them if you're into tinkering.

RAM caches on the drives themselves are a mixed bag. At moderate size (1−2MB), they are a potential big win for Windows 95/98, because Windows has stupid VM and I/O drivers. If you run a true multi−tasking OS like Linux, having unified RAM caches on the disk is a significant loss, since the overlapping I/O threads kick each other out of the cache, and the disk ends up performing work for nothing.

Most high−performance disks can be reconfigured (using mode page parameters, see above) to have `segmented' caches (sort of like a set−associative memory cache). With that configured properly, the RAM caches can be a moderate win, not because caching is so great on the disk (it's much better in the OS), but because it allows the disk controller more flexibility to reschedule its I/O request queue. You won't really notice it unless you routinely have >2 I/O requests pending at the SCSI level. The conventional wisdom (try it both ways) applies.

And finally I *do* have to make a disclaimer. Much of the stuff above is shameless simplification. In reality, high−performance SCSI disks are very complicated beasties. They run little mini−operating systems that are most comfortable if they have 10−20 I/O requests pending *at the same time*. Under those circumstances, the amortized global latencies are much reduced, though any single request may experience *longer* latencies than if it were the only one pending. The only really valid analysis are stochastic−process models, which we *really* don't want to get into here. :−)

4.9. Souping Up X Performance

If you care about X performance, be sure you get a graphics card with a dedicated blitter and a high−speed local−bus connection. If it says "AGP" you have this; AGP is a cross−vendor standard for a local bus optimized for graphics.

These cards speed up X in two ways. First, they offload some common screen−painting operations from the main processor onto specialized processors on the card itself. Secondly, by using a local bus, they make it possible to send commands to the card faster than the ISA bus could allow. The combined effect can be eye−poppingly fast screen updates even at very high resolutions.

There's no longer much reason to bother with any of the commercial X servers like MetroLink or X/Inside. XFree86 now supports most of the high−end cards that used to be the special preserve of the commercial X versions.

If you're feeling really flush, plump for a 15", 17" or even 20" monitor. The larger size can make a major difference in viewing comfort. Also you'll be set for 1600x1200, which many cards can support these days. In the mean time, the bigger screen will allow you to use fonts in smaller pixel sizes so that your text windows can be larger, giving you a substantial part of the benefit you'd get from higher pixel resolutions.

5. Hardware for Backups

You should have a tape drive for backup. Ideally, your tape backup should be able to image your entire disk. Choosing a tape drive used to be pretty complicated, with a plethora of different formats and media to chose from. It's much simpler now that the combination of cheap CD−ROM drives and huge hard disks has effectively killed off QIC and other sub−megabyte formats.

There are a bunch of non−tape niche technologies for backup, including floptical disks, Bernoulli boxes, Iomega and SyQuest removable drives, and magneto−optical drives. Ignore them all; they're half−assed attempts to combine a backup device with the fast random access needed for working storage that don't do either job very cost−effectively, especially when you consider the (high) cost of their media. Only magneto−optical drives are likely to have much of a future, and that only given improvements in access speed.

Digital Data Storage (DDS) capacities are a good match for today's multi−gigabyte drives (this is essentially the same technology as Digital Audio Tape or DAT). I'm told that Hewlett−Packard DDS devices are especially good, not surprising given HP's traditional obsession with reliability and overengineering stuff. All the DDSs I know about are SCSI devices.

At the high end, 8mm helical−scan tape (the stuff used in Sony camcorders) competes with DDS. This is a single−source tchnology, from Exabyte. Capacities are 2.2 and 5 gig, transfer speeds up around 500Kbytes/sec. However, a correspondent says ``Don't touch Exabyte. I've got three. All three have been sent back for warranty repair at least once.'' He also says ``A significant expense can be the cleaning tapes. Exabyte is notorious for this.'' So it's probably a good idea to stick with DDS if you have high−capacity requirements.

OTOH, Carl Renneberg <renneber@sci−log.apana.org.au> says of the Exabyte that his drive has proven to be rock steady and reliable, and recommends it with the following provisos:

- You *must*, simply *must*, use the correct kind of tape. Do not use 8mm video cartridges − they leave junk on the drive's head, and have dropouts. They are simply not designed for storing data. Use only the data grade tape from Sony (or from Exabyte − Sony manufactures the tape, Exabyte relabels it).
- Use the correct tape cleaner. Like tape cartridges, it's cheaper to buy the Sony brand than the Exabyte brand.

Carl has found that, where someone has had poor experience with Exabyte drives, it's because the owner − or the previous owner − did not take care of the drive, or used junk tapes.

Here's a quick summary of the major alternative DDS formats:

Table 2. DDS types

Type	Megabytes (uncompressed)	Megabytes (compressed)	Speed (Kbytes/sec)
DDS−1 60−meter	1300	2K−4K	183−366
DDS−1 90−meter	2000	4K−8K	183−366
DDS−2 120−meter	5000	7K−12K	183−500

DDS tape drives (and tapes) come actually in four variants: DDS, DDS−DC, DDS−2, and DDS−3. These are supposed to be downward compatible (e.g. DDS−2 reads/writes DDS−DC but not vice versa.) DDS and DDS−DC use 60m and 90m tapes; the −DC version adds hardware compression. DDS (non−DC) should be considered obsolete. DDS−2 adds 120m tapes and denser recording:

There is also a yet−newer DDS−3 standard, with yet again higher density on the tape. DDS−3 is bleeding edge (high premium), but DDS−2 is coming down now, and can make the difference between single−tape and and multi−tape backups (which can often make the difference between daily backups and "why didn't I..." hand−wringing.)

6. Of Mice And Machines

Mice and trackballs used to be simple; now, thanks to Microsoft, they're complicated. In the beginning, there was only the Mouse Systems 3−button serial mouse; this reported status to a serial port 30 times a second using a 5−byte serial packet encoding now called ``C'' protocol. The Logitech Series 7 and 9 mice were Mouse Systems−compatible. All Unixes that have any mouse support at all understand C−protocol serial mice.

Then Microsoft got into the act. They designed a two−button serial mouse which reports only deltas in a three−byte packet; that is, it sends changes in button status and motion reports only when the mouse is actually moving. This is called `M' protocol. Microsoft sold a lot of mice, so Logitech switched from `C' to `M'   but they added a third button, state changes for which show up in an optional fourth byte. Thus, `M+' protocol, upward−compatible with Microsoft's `M'. Most Unix vendors add support for M+ mice, but it's wise to check.

Bus mice are divided into 8255 and InPort types. These report info continuously at 30 or 60 Hz (though InPort mice have an option for reporting deltas only), and you get interrupts on events and then have to poll hardware ports for details. Bus mice are no longer widely available and there is no good reason to bother with them.

In addition to serial mice and bus mice, there are ``keyboard mice''. On PS/2s there are two identical−looking keyboard ports, labeled (with icons) ``mouse'' & ``keyboard''. Both are 6−pin mini−DINs that look like the regular PC keyboard port only smaller. Physically, the connector goes to the keyboard processor (often an 8042); electrically, it provides TTL levels (serial clock, serial data, ground and +5V); logically, it uses the same protocol as normal serial mice. The same keyboard processor that decodes the keyboard decodes the mouse. PS/2s have this port, many newer motherboards do as well.

All things considered, Unix users are probably best off going with a serial mouse if their motherboard isn't one of the newer ones with a built−in PS/2 mouse. Most current clone motherboards give you two serial ports, so you can still dedicate one to this and still have one for the all−important modem. Not only are the compatibility issues less daunting, but a serial mouse loads the multitasking system less due to interrupt frequency.

Beware that most clone vendors, being DOS oriented, bundle two−button M−type mice for which Unix support is presently spotty, and they may not work with your X. Thus, you may have to buy your own three−button mouse. Ignore the adspeak about dpi and pick a mouse/trackball that feels good to your hand.

Your editor really, really likes the Logitech TrackMarble, an optical trackball that eliminates the chronic roller−fouling problems of the older TrackMan. They're well−supported by XFree86 (type MouseMan), so any Linux or BSDI will accept them.

7. Modems

This section will give you a thumbnail sketch of the modem types available out there, one tuned for the typical Unix installation's needs.

7.1. Overview of the Modem Market

The modem market is like consumer electronics (and unlike the computer market as a whole) in that price is a very poor predictor of performance. For ordinary file transfers, some $50 modems are better than some $150 modems. Paying top dollar mainly buys you better tolerance of poor connections and better performance at heavy−duty bi−directional transfers (such as you would generate, for exmaple, using SLIP or PPP over a leased line to an Internet provider).

In today's market, the typical modem does a nominal 56kbps −− V.90 plus V.29 or V.17 fax transmission and reception. You don't see much in the way of slow/cheap to fast/expensive product ranges within a single brand, because competition is fierce and for many modem board designs (those featuring DSP (Digital Signal Processor) chips run by a program in ROM) adding a new protocol is basically a software change.

Detailed discussion of the V−series standards can be found in the Glossary. At this point all the jargon is mainly of historic interest.

For much more information on high−speed modems for Unix, see The Linux/Modem Compatibility Knowledge Base. The page is aimed at Linux users, but the advice is general to any non−Windows OS.

7.2. Internal vs. External

Most modems come in two packagings: internal, designed to fit in a PC card slot, and external, with its own case, power supply, and front−panel lights. Typically you'll pay $20 to $30 more for an external modem than you will for the internal equivalent. You'll also need a serial port to connect your external modem to.

Pay that premium   being able to see the blinkenlights on the external ones will help you understand and recover from pathological situations. For example, if your Unix system is prone to ``screaming−tty'' syndrome, you'll quickly learn to recognize the pattern of flickers that goes with it. Punch the hangup/reset button on an external modem and you're done   whereas with an internal modem, you have to go root and flounder around killing processes and maybe cold−boot the machine just to reset the card.

See Rick's Rants for extended discussion of this point.

7.3. Pitfalls to Avoid

Beware of hype. Some modems with 14400 or higher FAX transmission rates have a lower speed−limit on data transmissions. If you're in any doubt, get a quote of the ``data rate'', not FAX bit rate.

The best way to avoid problems is to make the vendor tell you what V−series standards the product supports. Then use the handy glossary above.

If the abbreviation ``RPI'' occurs anywhere on the box, don't even consider buying the modem. RPI (Rockwell Protocol Interface) is a proprietary ``standard'' that allows modem makers to save a few bucks at your expense by using a cheap−jack Rockwell chipset that doesn't do error correction. Instead, it hands the job off to a modem driver which (on a Unix machine) you will not have.

Also avoid anything called a ``Windows Modem'' or ``WinModem'', ``HCF, or ``HSP''; these lobotomized pieces of crap require a Windows DLL to run. They will eat up to 25% of your processor clocks during transfers, and hog high−priority interrupts (causing your machine to stall under Windows even if your processor still has spare cycles).

All modems advertise mainly on DCE speed. Get a quote for max DTE speed too, if you can; make sure the DTE rate is enough times faster to handle your maximum on−the−fly compression (with V.42bis that's 4:1).

Multi−user Unix eats enough processor clocks that you want to be sure of good hardware buffering in your UART   that is, enough of it to avoid losing characters between modem and PC if the OS is a bit slow responding to an interrupt (V.42bis in hardware won't detect this!). This means you want a 16550A or equivalent UART. If you're using an external modem, this is an issue about your serial−port board(s). If you're using an internal modem, the UART is on the modem card itself. So, when buying internal modems, *ask* what the UART type is. If the vendor says 16540, lose them.

Many fax modems come with bundled MS−DOS fax software that is at best useless under Unix, and at worst a software kluge to cover inadequate hardware. Avoid these bundles and buy a bare modem   it's cheaper, and lowers the likelihood that something vital to your communications needs has been left out of the hardware.

Avoid ``Class 1'' and ``Class 2'' modems. Look for ``Class 2.0'' for the full EIA−standard command set.

7.4. A Modem Glossary

An explanation of modem jargon. None of this is very interesting any more in the age of ubiquitous cheap V.90 modems.

First, some ancient history:

Bell 103

AT&T 110bps standard. Dead as a doornail. Anyone tries to sell you a Bell 103 modem, report 'em to the bunco squad. The damn things were the size of breadboxes and flakier than bad dandruff. Not totally useless, though; they make decent space heaters.

Bell 202

AT&T standard for 300bps transmission. Every modem in existence does this one (or, at least, if you find one that doesn't, donate it to an archeological museum instantly).

Bell 212A

AT¦T standard for 1200bps transmission. Yes, a few people are still chugging gamely away with 1200bps modems.

MNP 2, MNP 3, MNP 4

Microcom Network Protocol, a series of error−detection−and−correction standards primarily used for 1200 and 2400−baud modems. Now incorporated in V.42.

MNP 5

Microcom Network Protocol Level 5, a an error−correction and data−compression method that runs on top of MNP 2−4, now displaced by V.42bis. (Most modems that support V.42bis also support the MNP 5 and auto−detect it in use by other modems).

Now a new term enters the picture:

CCITT

International Consultative Committee on Telephony and Telegraphy (the acronym is from the French official name of the organization, Comitè Consultatif Internationale Tèlègraphique et Tèlèphonique). This is the standards body for modems, fax equipment and related gear. Now known as ITU−T (International Telecommunications Union− Telecommunication Standards Sector).

ITU−T is responsible for the "V" series of modem specifications, These are:

V.22

ITU−T standard for 1200bps. Not quite compatible with Bell 212A, thus never used much in the U.S. This caused a few trans−border snafus in the early 1980s. Most modems still offer a V.22 compatibility mode, but nobody doing international data transfers settles for 1200bps any more, so it's a dead issue.

V.22bis

ITU−T standard for 2400bps. AT&T got out of the modem−standards business after 212A; thus, V.22 was the first open standard to be supported worldwide.

V.25ter

ITU−T standard for the AT command set.

V.32

The ITU−T standard for 9600bps transmission.

V.32bis

The ITU−T standard for 12200 and 14400bps data transmission.

V.42

The ITU−T standard for firmware error correction between V.32 and V.32bis modems (also worked with V22 and V22bis). Helps them cope with noisy lines.

V.42bis

The ITU−T standard for V.42 with on−the−fly compression. Same bit speeds as V.32bis and V.42, but a higher effective transfer rate (especially for text, graphic images and other high−redundancy data, for which it may manage up to 4:1 compression).

V.34

ITU−T standard for 28800bps transmission. Also known as V.fast before the standard was ratified in June 1994. Amended in 1996 to support 33600bps transmission.

V.90

ITU−T recommendation for asymmetric data signalling rates of up to 56Kbps in the direction of a digitally connected server to a capable client, and up to 33.6Kbps in the direction of the client to the server.

The technology is based on eliminating restrictions imposed by the conversion of analog signals to digital form in the downstream data path (server −> client). Data flow in the server to client direction does not occur in the form of a modulated carrier, it is instead sent as binary numbers representative of 256 possible voltage levels. The reason for the asymmetrical send/receive rates is because in the direction from the client to the server it is not possible to use a digital coding scheme and make it work as well as V.34 does, thus V.34 is used instead. It isn't possible because the telco's line card has a codec that is a much better digital level changer for the transmit direction than it is for the receive direction. The codec used in the customer's modem is, in that respect, somewhat more sophisticated and was designed to work as a fairly good level changer in the receive direction (which the telco's codec was not designed to do).

Note: Achievable bit rates are limited to less than 56kbps in the United States by FCC regulations that limit power input to the network.

These ITU−T protocols form a neat speed and capability ladder. Usually, any modem that can do V.32bis can do V.32; any modem that can do V.42 can do V.32bis; any modem that can do V.42bis can do V.42; any modem that can do V.34 can do V.32; and so forth. In fact, modems in this family automatically recognize each other and negotiate for the highest speed both can handle.

Beyond the V series, the telcos are pushing a technology called ADSL (Asymmetric Digital Subscriber Line). This method converts existing twisted−pair telephone lines into access paths for multimedia and high speed data communications. ADSL transmits more than 6Mbps to a subscriber, and as much as 640 kbps more in both directions.

An ADSL circuit connects an ADSL modem on each end of a twisted−pair phone line, creating three information channels; a high speed downstream channel, a medium speed duplex channel, and a POTS (Plain Old Telephone Service) channel. The POTS channel is split off from the digital modem by filters, thus guaranteeing uninterrupted POTS, even if ADSL fails. The high speed channel ranges from 1.5 to 6.1 Mbps, while duplex rates range from 16 to 640 kbps. Each channel can be sub−multiplexed to form multiple, lower rate channels.

Now to the vendor−proprietary encodings:

V.32terbo

A non−ITU−T ``standard" for 16800 and 19200bps transmission floated by a vendor consortium frustrated with the glacial pace of the V.fast development. V.terbo was intended as a stopgap until V.fast comes in; technically it's very close to V.32bis. Obsolete since V.34 was ratified.

HST

Proprietary 16.8kcps protocol used by US. Robotics modems. Never very important in the Unix world, and now being eclipsed by the V−series modems.

PEP

Packet Exchange Protocol, used for 19200bps between Telebit modems. These modems were very widely used in the Unix/USENET/UUCP community at one time (for years, Telebit offered substantial discounts and/or two−for−one deals to buyers with Internet or UUCP addresses). They were super−reliable, programmable, fast as all get out, famously good at pushing bits through noisy and lossy phone lines, and died stone dead because they were a single−source product designed around a proprietary protocol (and correspondingly expensive).

TurboPEP

Enhanced PEP, 115200bps   faster than V.fast even dreams of (but don't expect to get quite that full speed unless your connection is *really* clean).

56Kbps Modems

[Pre−V.90] − Rockwell, USR, Lucent Technologies, and Motorola marketed incompatible chipsets/modems that operated in a server/client format at up to 56Kbps over standard telephone lines prior to the adoption of ITU−T V.90. USR implemented a protocol dubbed X2, and the remainder combined efforts to implement a protocol dubbed K56Flex (a combination of Rockwell's K56Plus and Lucent's VFlex/2 protocols). The X2 and K56Flex protocols do not interoperate.

Finally, some important modem features:

DCE

Data Communications Equipment rate, the maximum modem−to−modem bit speed.

DTE

Data Terminal Equipment rate, maximum computer−to−modem bit speed. If your modem does N−to−1 compression, be sure your DTE rate is at least N times your DCE rate, or you won't be able to keep the modem fed fast enough to get top DCE speed.

And one important misfeature:

RPI

Rockwell Protocol Interface −− a scam designed to save manufacturer a few bucks by omitting error−correction logic from V−series modems. Instead it's done in software by a proprietary driver. Do not buy such a modem, it won't fly under Unix. See the Pitfalls to Avoid section.

Fax capability is included with most all modems these days; it's cheap for manufacturers, being basically a pure software add−on. The CCITT also sets fax protocol standards. Terms to know:

V.29

CCITT standard for Group III fax encoding at 9600bps

V.17

CCITT standard for Group III fax encoding at 14400bps

V.17 isn't common yet, but it doesn't usually cost extra over V.29 when you find it.

There's a separate series of standards for software control of fax modems over the serial line maintained by the Electronics Industry Association and friends. These are:

Class 1 base EIA standard for fax control as extensions to the Hayes AT command set.

Class 2.0 enhanced EIA standard including compression, error correction, station ID and other features.

Class 2 marketroidian for anything between Class 1 and Class 2.0. Different ``Class 2'' modems implement different draft subsets of the 2.0 standard, so ``Class 2'' fax software won't necessarily drive any given ``Class 2'' modem.

There's also a proprietary Intel "standard" called CAS, Communicating Applications Specification. Ignore it; only Intel products support it.

The GNU toolset includes a freeware fax transmission and reception toolset, Netfax. Look for it at prep.ai.mit.edu:pub/gnu/fax−*. It says it requires a modem conforming to the ``Class 2'' control standard, but you'd be safest getting a 2.0−conformant modem for reasons explained above. Netfax also requires GNU Ghostscript to do Postscript handling for it.

8. CD−ROMs and Multimedia Hardware

8.1. CD−ROM Drives

Standard CD−ROMs hold about 650 megabytes of read−only data in a format called ISO−9660 (formerly ``High Sierra''). All current Unixes now support these devices. In fact, most Unix and Linux software is now distributed on ISO−9660 CD−ROM, a cheaper and better method than the QIC tapes we used to use.

CD−ROM drives may be driven through SCSI, through enhanced IDE (ATAPI), or through a proprietary interface card (like the Mitsumi and Sony interfaces). Unix support for CD−ROMs is usually through SCSI drivers (exception: BSDI/386 supports the Mitsumi interface; Linux supports the Mitsumi and Sony interfaces). I recommend that you avoid the proprietary cards; they will effectively cost you money when you need to upgrade. Besides, street prices for SCSI CD−ROMS have dropped below $200.

(A few external CD−ROMs come with a parallel−port interface. Avoid these; they tend to have very slow transfer rates.)

Any CD−ROM you buy should be at least a ``double−spin'' drive meeting the MPC2 (Multimedia PC) standard of a 300K/sec transfer rate when reading. digital data. The older single−speed drives, which only supported the 150K/sec rate Red Book standard for audio CDs, are obsolete. The lowest speed you can buy these days is 4X (600K/sec). 6X, 8X, 10X, 12X, 24X, and 32X are available.

The next level up in CD hardware standards is CD−ROM XA. So far, drives that support XA are few and expensive. It's not yet in wide use in the DOS/Windows world, and I don't know of any Unix support for it, either in commercial or freeware code.

CD−ROM access times about 280ms for high−end double−speed drives (to put this in perspective, it's about 30 times slower than a typical 9ms hard disk, but considerably faster than a tape). Accordingly, modern 32X drives are about half the speed of a hard drive.

Many CD−ROM drives use a caddy (a plastic frame into which you drop the CD before inserting both in the drive slot). Some are caddyless. The caddy has two advantages: (1) it helps protect the CD from scratches, and (2) it pre−positions the and supports the CD for the drive spindle, allowing faster controlled rotation. Consequently, most of the faster CD−ROM drives use caddies. They have the disadvantage of making CD−changing slightly more awkward.

Most CD−ROMS will include a headphone jack so you can play audio CDs on them. Better−quality ones will also include two RCA jacks for use with speakers. Another feature to look for is a drive door or seal that protects the drive head from dust.

CD−ROM formats are still an area of some confusion. A slight enhancement of the original ``High Sierra'' CD−ROM filesystem format (designed for use with DOS, and limited to DOS's 8+3 file−naming convention) has been standardized as ISO−9660.

There is a de−facto Unix standard called `Rock Ridge' pioneered by the Sun User's Group shareware CD−ROMs. This is a way of putting an extra layer of indirection on an ISO−9660 layout that preserves Unix's long dual−case filenames. Some Unixes (notably Linux, netBSD, freeBSD and BSD/OS) can mount Rock Ridge filesystems.

More much more detail on CD−ROMs, CD−ROM standards and how to buy drives is available in the alt.cdrom FAQ, available for FTP as cdrom.com:/cdrom/faq. It is also archived in the news.answers tree at rtfm. This FAQ includes comparison tables tables of numerous drive types, CD−ROM sources, and ordering information.

8.2. Sound Cards and Speakers

Look for the following features as a minimum in your sound card:

- 16−bit sampling (for 16,5536 dynamic levels rather than 256).
- Mono and stereo support.
- Sampling rates ranging fron 8K/sec (voice−quality) through 11KHz (AM−radio quality), 22KHz (FM−radio quality) and standard audio (44.1KHz).
- MIDI interface via a standard 15−pin D−shell connector.
- RCA output jacks for headphones or speakers.
- A microphone jack for sound input.

The most important feature is your sound card is what type of MIDI synthesis it has. All current sound cards have sort of MIDI (musical instrument digital interface) compliance, but if nothing else, make sure your card supports the General MIDI standard. MIDI is just a set of commands issued by the application that tells the sound card which instrument to play, at what note, and for what duration.

Older and cheaper cards use FM synthesis. This synthesis uses a combination of sine waves to imitate the sounds of the different instruments. The result is like the sound tracks of most computer games sold a few years ago; imitation music with an arcade−like sound.

The method used by most modern sound boards is called wave table synthesis. In this method, digitized samples of actual instrument sounds are used as templates for the tones generated by the MIDI commands. Wave table cards vary in the quantity and quality of samples; one figure of merit often quoted is the wave table ROM size (often 4MB or 8MB). Also some boards have wavetable RAM that can store samples loaded from a disk.

Soundcards with DSP (Digital Signal Processing) can perform synthesis effects on board, relieving the CPU for other tasks. Some DSP chips are even software−programmable. Some high−end cards even include 3D sound effects. Whether the system used is SRS (Sound Retrieval System), Q−Sound, or Spatializer, it is designed to improve the perceived stereo effect of your speakers. These 3D effects work by delaying the timing of certain portions of the audio signal so that different frequencies hit your ear at slightly different times. The downside is that some of the cards equipped with 3D sound add a noticeable amount of noise to the card's output.

If you play a lot of computer games, you'll need to pay attention to compatibility. DOS games are written almost exclusively for the Creative Labs specification; you will need a card that is 100% Sound Blaster compatible. Many vendors do not license the Creative Labs specification but claim that their cards are 100% game compatible. This means that the sound will work, but not all sounds that you hear will be the ones that the game programmers intended. If you play many DOS games, it would be best to buy a Sound Blaster and save yourself a migraine.

Lastly, try to avoid sound cards with built−in amplifiers that are more powerful than 4 watts/channel. Sound cards that have more powerful amplifiers are said to have the problem of adding noise to the card's output.

Use powered speakers with a 4 watt/channel card to solve this problem. Most cards are equipped with 4 watts/channel anyway. Wavetable cards are so inexpensive these days that it's almost worth their additional cost over a regular FM synthesis card. If you decide to settle for an FM card, make sure that there is a daughterboard made for the card that will let you upgrade to wavetable synthesis. In some cases, however, the wavetable card is cheaper than buying an FM card and then deciding that you want the wavetable upgrade. If you do decide on the wavetable as your card of choice, PC Magazine rated the best MIDI wavetables (MIDI being the most important feature in my opinion) the Media Vision Premium 3−D, Media Vision Pro 3−D, Creative Labs Sound Blaster AWE32, and the Turtle Beach Monterey (although there are value editions of the Sound Blaster 32 that have fewer ROM instrument samples but maintain the superior MIDI wavetable synthesis).

Linux, includes drivers for the SoundBlaster series and many other boards including the PAS Adlib, the Gravis Ultrasound, and the ProAudioSpectrum. Many of these are included in the 2.0 and newer kernels. See the OSS/Free home page for details

Warning: some sound cards require a specific CD−ROM type! Avoid these.

In speakers, look for a magnetically−shielded enclosure with volume, bass and treble controls. Some speakers run off the card's 4−watt signal; others are ``self−powered'', using batteries or a separate power supply. Your major buying choice is which one of these options to pursue.

If you're interested in music composition and playback (as opposed to just sound effects) another important buying choice is FM versus wave−table synthesis. FM is cheaper, wave−table is better. In−depth discussion of this tradeoff is beyond the scope of this document. For more details, see the PCsoundboards/generic FAQ (available on rtfm in the news.answers archive).

Generally speaking, you'll get better value if you buy your sound card, CD−ROM, and speakers not separately but as a bundled ``multimedia upgrade kit'' (even though Unix users will usually have to throw out the bundled software).

One final, important tip: that audio cable from your CD−ROM back to the sound card is used only when you play audio CD−ROMs through your speakers. Software−generated sound goes through the system bus, so you can play ``Doom'' with sound even if your sound board won't accept the audio cable connector.

9. Special considerations when buying laptops

Up until about 1999 the laptop market was completely crazy. The technology was in a state of violent flux, with ``standards'' phasing in and out and prices dropping like rocks. Things are beginning to settle out a bit more now.

One sign of this change is that there are now a couple of laptop lines that are clear best−of−breeds for reasons having as much to do with good industrial design and ergonomics as the technical details of the processor and display.

In lightweight machines, I'm a big fan of the Sony VAIO line. I've owned one from early 1999 until it physically disintegrated under the rigors of travel in late 2000, and could hardly imagine switching. They weigh 3.5 pounds, give you an honest 3 hours of life per detachable battery pack, have a very nice 1024x768 display, and are just plain *pretty*. Their only serious drawback is that they're not rugged, and often fall apart after a year or so of use.

If you want a full−power laptop that can compete with or replace your desktop machine, the IBM ThinkPad line is the bomb. Capable, rugged, and nicely designed (though somewhat heavyweight for my taste).

These machines are not cheap, though. If you're trying to save money by buying a no−name laptop, here are things to look for:

First: despite what you may believe, the most important aspect of any laptop is *not* the CPU, or the disk, or the memory, or the screen, or the battery capacity. It's the keyboard feel, since unlike in a PC, you cannot throw the keyboard away and replace it with another one unless you replace the whole computer. *Never buy any laptop that you have not typed on for a couple hours*. Trying a keyboard for a few minutes is not enough. Keyboards have very subtle properties that can still affect whether they mess up your wrists.

A standard desktop keyboard has keycaps 19mm across with 7.55mm between them. If you plot frequency of typing errors against keycap size, it turns out there's a sharp knee in the curve at 17.8 millimeters. Beware of ``kneetop'' and ``palmtop'' machines, which squeeze the keycaps a lot tighter and typically don't have enough oomph for Unix anyway; you're best off with the "notebook" class machines that have full−sized keys.

Second: be careful that your laptop meets the minimum core and disk requirements for the Unix you want to run. This is generally not a problem with desktop machines, which can be upgraded cheaply and easily, but laptops often have more stringent constraints.

Third: with present flatscreens, 1024x768 color is the best you're going to do (though that may change soon). If you want more than that (for X, for example) you have to either fall back to a desktop or make sure there's an external−monitor port on the laptop (and many laptops won't support higher resolution than the flatscreen's).

Fourth: look for Nickel−Metal−Hydride (NiMH) batteries, as opposed to the older (Nickel−Cadmium) NiCad type. NiMH batteries are great because they have considerably higher energy capacity per pound that NiCads. They need special circuitry to charge them fast, so don't try to throw out your NiCads and replace them with NiMH cells if you use a fast charger intended for NiCads. Both kinds of cells can be damaged by overcharging at rates faster than 10 hours per full charge.

Fifth: Older laptop electronics are still 5−volt CMOS. Most current designs are 3.3−volt CMOS with power−management features on the processor (these are often labelled APM, Advanced Power

Management). Buy this, if you can, to nearly double your use time between recharges.

Sixth: about those vendor−supplied time−between−recharge figures; *don't believe them*. They collect those from a totally quiescent machine, sometimes with the screen or hard disk turned off. Under DOS, you'd be lucky to get half the endurance they quote; under Unix, which hits the disk more often, it may be less yet. Figures from magazine reviews are more reliable.

Seventh: You probably want a color dual−scan display. It used to be that you had to choose between passive−matrix LCD (cheap, miserable color) and active− matrix LCD (great color, horribly expensive). Dual−scan passive−matrix is nearly as good as active−matrix, except for the narrower viewing angle, and it's much cheaper. Avoid the older single−scan models, sometimes marketed as having STN (super−twisted nematic) displays.

Eighth: get either a CD−ROM drive or an Ethernet card. Otherwise initial load of your Unix could turn into a serious problem...

10. How to Buy

10.1. When to Buy

It used to be that good configurations for Unix were what the market called `server' machines, with beefed−up I/O subsystems and fast buses. No longer; today's `servers' are monster boxes with multiple power supplies and processors, gigabytes of memory, and industrial−grade air cooling −− they're not really suitable as personal machines. A typical SCSI desktop workstation is as much as you'll need.

Prices keep dropping, so there's a temptation to wait forever to buy. A tactic that makes a lot of sense in this market, if you have the leisure, is to fix in your mind a configuration and a trigger price that's just a little sweeter than the market now offers and buy when that's reached.

Before you shop, do your homework. Publications like "Computer Shopper" (and their web site at http://www.computershopper.com) are invaluable for helping you get a feel for prices and what clonemakers are doing. Another excellent site is ComputerESP.

10.2. Where to Buy

The most important where−to−buy advice is negative. Do *not* go to a traditional, business−oriented storefront dealership. Their overheads are high. So are their prices.

Especially, run −− do not walk −− away from any outfit that trumpets `business solutions'. This is marketing code for the kind of place that will justify a heavy price premium by promising after−sale service and training which, nine times out of ten, will turn out to be nonexistent or incompetent. Sure, they'll give you plush carpeting and a firm handshake from a guy with too many teeth and an expensive watch −− but did you really want to pay for that?

There are two major alternatives to storefront dealerships and one minor one. The major ones are mail order and computer superstores. The minor one is computer fairs.

10.3. Computer Fairs

I used to be a big fan of hole−in−the−wall stores run by immigrants from the other side of the International Date Line, but most of those places have been driven out of the regular retail game by the superstores. The only place you find diaspora Chinese and Indians selling cheap PCs over the counter anymore is at computer fairs. (Usually they're doing it to publicize a mail−order business.)

You can find good `loss−leader' deals on individual parts at these fairs (they're especially good places to buy disk drives cheap). But I call them a minor alternative because it's hard to get a custom SCSI−based configuration tuned for Unix built for you at a fair. So you end up, effectively, back in the mail−order or Web channel.

10.4. Mail Order

Direct−mail or Web buying makes a lot of sense today for anyone with more technical savvy than J. Random Luser in a suit. Even from no−name mail−order houses, parts and system quality tend to be high and consistent, so conventional dealerships don't really have much more to offer than a warm fuzzy feeling. Furthermore, competition has become so intense that even mail−order vendors today have to offer not just lower prices than ever before but warranty and support policies of a depth that would have seemed incredible a few years back. For example, many bundle a year of on−site hardware support with their medium− and high−end "business" configurations for a very low premium over the bare hardware.

Note, however, that assembling a system yourself out of mail−order parts is *not* likely to save you money over dealing with the mail−order systems houses. You can't buy parts at the volume they do; the discounts they command are bigger than the premiums reflected in their prices. The lack of any system warranty or support can also be a problem even if you're expert enough to do the integration yourself   because you also assume all the risk of defective parts and integration problems.

Cruise through `Computer Shopper' and similar monthly ad compendia. Even if you decide to go with a conventional dealer, this will tell you what *their* premiums look like.

Watch out for dealers (Spectrum Trading for one) who charge ridiculous shipping fees. One of our spies reports he bought a hotswappable hard disc drive tray that weighed about 3 lbs. and cost $250 and they charged $25 to ship it UPS groud.

Don't forget that (most places) you can avoid sales tax by buying from an out−of−state mail−order outfit, and save yourself 6−8% depending on where you live. If you live near a state line, buying from a local outfit you can often win, quite legally, by having the stuff shipped to a friend or relative just over it. Best of all is a buddy with a state−registered dealer number; these aren't very hard to get and confer not just exemption from sales tax but (often) whopping discounts from the vendors. Hand him a dollar afterwards to make it legal.

(Note: I have been advised that you shouldn't try the latter tactic in Florida −− they are notoriously tough on "resale license" holders).

(Note II: The Supreme Court has ruled that states may not tax out−of−state businesses under existing law, but left the way open for Congress to pass enabling legislation. Let's hope the mail−order industry has good lobbyists.)

If you can afford to trade a little extra money for minimum hassles, you should go with a Linux systems integrator like VA Linux Systems. These guys will sell you tested and tuned hardware with Linux pre−loaded. What you get for your price premium includes a support line to call.

10.5. Computer Superstores

Big chain superstores like CompUSA and Circuit City give you a reasonable alternative to mail order. And there are good reasons to explore it −− these stores buy and sell at volumes that allow them to offer prices not far above mail−order.

One thing you should not buy mail−order if you can avoid it is a monitor. Monitors are subject to substantial quality variations even within the same make and model. Also, one good bump during shipping can twist the yoke on a monitor so the image is tilted with respect to the bezel. So buy your monitor face−to−face, picking the best out of three or four.

Another good argument for buying at a superstore is that you may have to pay return postage if you ship a mail−order system back. On a big, heavy system, this can eat your initial price savings.

The only major problem with superstores is that the salespeople who staff them aren't very bright or very clueful (it's a sort of Darwinian reverse−selection effect; these are the guys who are fascinated by computer technology but not smart enough to be techies). Most of them don't know from Linux and are likely to push things like Plug'n'Play cards and two−button mice and (worse!) controllerless modems, that you can't use. Use caution and check your system manifest.

10.6. Other Buying Tips

You can often get out of paying tax just by paying cash, especially at computer shows. You can always say you're going to ship the equipment out of the state.

A lot of vendors bundle DOS or Windows and variable amounts of apps with their hardware. If you tell them to lose all this useless cruft they may shave $50 or $100 off the system price.

11. Questions You Should Always Ask Your Vendor

11.1. Minimum Warranty Provisions

The weakest guarantee you should settle for in the mail−order market should include:

- 72−hour burn−in to avoid that sudden infant death syndrome. (Also, try to find out if they do a power−cycling test and how many repeats they do; this stresses the hardware much more than steady burn−in.)
- 30 day money−back guarantee. Watch out for fine print that weakens this with a restocking fee or limits it with exclusions.
- 1 year parts and labor guarantee (some vendors give 2 years).
- 1 year of 800 number tech support (many vendors give lifetime support).

Additionally, many vendors offer a year of on−site service free. You should find out who they contract the service to. Also be sure the free service coverage area includes your site; some unscrupulous vendors weasel their way out with "some locations pay extra", which translates roughly to "through the nose if you're further away than our parking lot".

If you're buying store−front, find out what they'll guarantee beyond the above. If the answer is "nothing", go somewhere else.

11.2. Documentation

Ask your potential suppliers what kind and volume of documentation they supply with your hardware. You should get, at minimum, operations manuals for the motherboard and each card or peripheral; also an IRQ list. Skimpiness in this area is a valuable clue that they may be using no−name parts from Upper Baluchistan, which is not necessarily a red flag in itself but should prompt you to ask more questions.

11.3. A System Quality Checklist

There are various cost−cutting tactics a vendor can use which bring down the system's overall quality. Here are some good questions to ask:

- Is the memory zero−wait−state? One or more wait states allows the vendor to use slower and cheaper memory but will slow down your actual memory subsystem throughput. This is a particularly important question for the cache memory!
- If you're buying a factory−configured system, does it have FCC certification? While it's not necessarily the case that a non−certified system is going to spew a lot of radio−frequency interference, certification is legally required and becoming more important as clock frequencies climb. Lack of that sticker may indicate a fly−by−night vendor, or at least one in danger of being raided and shut down! (For further discussion, see the section on Radio Frequency Interference above.)

- Are the internal cable connectors keyed, so they can't be put in upside down? This doesn't matter if you'll never, ever *ever* need to upgrade or service your system. Otherwise, it's pretty important; and, vendors who fluff this detail may be quietly cutting other corners.

---

12. Things to Check when Buying Mail−Order

12.1. Tricks and Traps in Mail−Order Warranties

Reading mail−order warranties is an art in itself. A few tips:

Beware the deadly modifier ``manufacturer's'' on a warranty; this means you have to go back to the equipment's original manufacturer in case of problems and can't get satisfaction from the mail−order house. Also, manufacturer's warranties run from the date *they* ship; by the time the mail−order house assembles and ships your system, it may have run out!

Watch for the equally deadly ``We do not guarantee compatibility''. This gotcha on a component vendor's ad means you may not be able to return, say, a video card that fails to work with your motherboard.

Another dangerous phrase is ``We reserve the right to substitute equivalent items''. This means that instead of getting the high−quality name−brand parts advertised in the configuration you just ordered, you may get those no−name parts from Upper Baluchistan   theoretically equivalent according to the spec sheets, but perhaps more likely to die the day after the warranty expires. Substitution can be interpreted as ``bait and switch'', so most vendors are scared of getting called on this. Very few will hold their position if you press the matter.

Another red flag: ``Only warranted in supported environments''. This may mean they won't honor a warranty on a non−DOS system at all, or it may mean they'll insist on installing the Unix on disk themselves.

One absolute show−stopper is the phrase ``All sales are final''. This means you have *no* options if a part doesn't work. Avoid any company with this policy.

12.2. Special Questions to Ask Mail−Order Vendors Before Buying

- Does the vendor have the part or system presently in stock? Mail order companies tend to run with very lean inventories; if they don't have your item in stock, delivery may take longer. Possibly *much* longer.
- Does the vendor pay for shipping? What's the delivery wait?
- If you need to return your system, is there a restocking fee? and will the vendor cover the return freight? Knowing the restocking fee can be particularly important, as they make keep you from getting real satisfaction on a bad major part. Avoid dealing with anyone who quotes more than a 15% restocking fee   and it's a good idea, if possible, to avoid any dealer who charges a restocking fee at all.

Warranties are tricky. There are companies whose warranties are invalidated by opening the case. Some of those companies sell upgradeable systems, but only authorized service centers can do upgrades without invalidating the warranty. Sometimes a system is purchased with the warranty already invalidated. There are vendors who buy minimal systems and upgrade them with cheap RAM and/or disk drives. If the vendor is not an authorized service center, the manufacturer's warranty is invalidated. The only recourse in case of a problem is the vendor's warranty. So beware!

12.3. Payment Method

It's a good idea to pay with AmEx or Visa or MasterCard; that way you can stop payment if you get a lemon, and may benefit from a buyer−protection plan using the credit card company's clout (not all cards offer buyer−protection plans, and some that do have restrictions which may be applicable). However, watch for phrases like ``Credit card surcharges apply'' or ``All prices reflect 3% cash discount'' which mean you're going to get socked extra if you pay by card.

Note that many credit−card companies have clauses in their standard contracts forbidding such surcharges. You can (and should) report such practices to your credit−card issuer. If you already paid the surcharge, they will usually see to it that it is returned to you. Credit−card companies will often stop dealing with businesses that repeat such behavior.

12.4. Which Clone Vendors to Talk To

12.4.1. Some picks

If you've got the bucks to pay for high−end, high−quality hardware, I don't know of anybody better than VA Linux Systems. (Full disclosure: They've given me hardware and made me a director of the company −− but all long after I got real happy with the way they do things.)

12.4.2. Some pans

Dell: treated the Unix community, customers and its own employees very badly back in 1994 by making an internal decision to kill its market−leading SVr4 port, then obfuscating and lying about its intentions for months after its actions made the direction clear. On the other hand, they started getting serious about Linux in early 2000. I won't say boycott them the way I used to, but I will say you ought to think about VA instead.

Gateway: may also be a vendor to avoid. Apparently their newer machines don't have parity bits in their memories; memory is tested only on reboot. This is dubious design even for DOS, and totally unacceptable for Unix.

13. Software to go with your hardware

I used to maintain an entire separate FAQ on Unixes for 386/486 and Pentium hardware. Times change, industries evolve, and I can now replace that FAQ with just three words:

Go get Linux!

14. Other Resources on Building Linux PCs

Andrew Comech's The Cheap /Linux/ Box page is a useful guide to building with current hardware that is updated every two weeks. Andrew also maintains a short−cut version.

The Caveat Emptor guide has an especially good section on evaluating monitor specifications.

Dick Perron has a PC Hardware Links page. There is lots and lots of good technical stuff linked to here. Power On Self Test codes, manufacturer address lists, common fixes, hard disk interface primer, etc.

Anthony Olszewski's Assembling A PC is an excellent guide to the perplexed. Not Linux−specific. If you're specifically changing a motherboard, see the Installing a Motherboard page. This one even has a Linux note.

Tom's Hardware Guide covers many hardware issues exhaustively. It is especially good about CPU chips and motherboards. Full of ads and slow−loading graphics, though.

The System Optimization Site has many links to other worthwhile sites for hardware buyers.

Christopher B. Browne has a page on Linux VARs that build systems. He also recommends the Linux VAR HOWTO.

Jeff Moe has a Build Your Own PC page. It's more oriented towards building from parts than this one. Less technical depth in most areas, but better coverage of some including RAM, soundcards and motherboard installation. Features nifty and helpful graphics, one of the better graphics−intensive pages I've seen. However, the hardware−selection advice is out of date.

The Linux Hardware Database provides, among other things (e.g., drivers, specs, links, etc.), user ratings for specific hardware components for use under Linux. Our ratings take a lot of the guess work out of choosing which hardware to buy for a Linux box. The site also provides several product−specific resources (i.e., drivers, workarounds, how−to) that help users get hardware working after they have made a purchase.

15. Upgrading Older Machines

15.1. Older Memory Types

The last−generation memory package was a SIMM (Single In−Line Memory Module). We include some information on these here for people upgrading old systems.

SIMMs have 9 bits wide (8 bits of data, one for parity) and either 1, 4, or 16 megabytes in size. The 16s are slightly cheaper per megabyte.

There are two physical SIMM sizes; 30−pin and 72−pin. The 30−pin size is long obsolete; most SIMM−using motherboards have exclusively 72−pin sockets.

Also, your memory may or may not have on−board parity check bits. Memory with parity is specified as ``x36'' (4 check bits per word); memory without is ``x32''. Some motherboards require parity memory; some (including those based on the Neptune chipset) can use parity if present; some (including those based on the Triton chipset) cannot use it. Parity memory is a good idea for Unix machines, which stress the hardware more than other OSs.

You'll hear a lot about ``EDO'' (Extended Data Out) RAM. This stuff is 5%−10% more expensive, but that pays for a significant performance boost in ``burst mode'' (that is, when fetching large contiguous sections of memory into the cache). Most current motherboards (including those based on the Triton chipset) can use EDO but don't require it.

You do *not* want to buy 1MB SIMMs any more, unless you're filling out memory in an existing system configured with 1MB SIMMs. The trouble with these (besides the fact that four of them are more expensive than a 4MB chip) is that, due to interleaving constraints, you'll probably have to throw them away if you ever add more memory.

These interleaving constraints differ from motherboard to motherboard. Typically, memory sockets are set up in banks of either 2 or 4, with a requirement that you fill all sockets of each bank with the same size of chip. Also, if you mix chip sizes in the banks, the sizes have to decrease smoothly (so there won't be gaps in the memory address space).

So your possible configurations look like the following table. All of these are possible on 4−bank, 16−slot boards; only the asterisked ones are possible on 2−bank, 8−slot boards. All the capacities above 64MB require that the board accept 16MB SIMMs. All current 2x8 boards accept these.

```
        * 4MB  = 1−1−1−1
        * 8MB  = 1−1−1−1 + 1−1−1−1
          12MB = 1−1−1−1 + 1−1−1−1 + 1−1−1−1
          16MB = 1−1−1−1 + 1−1−1−1 + 1−1−1−1 + 1−1−1−1

        * 16MB = 4−4−4−4
        * 20MB = 4−4−4−4 + 1−1−1−1
          24MB = 4−4−4−4 + 1−1−1−1 + 1−1−1−1
          28MB = 4−4−4−4 + 1−1−1−1 + 1−1−1−1 + 1−1−1−1

        * 32MB = 4−4−4−4 + 4−4−4−4
          36MB = 4−4−4−4 + 4−4−4−4 + 1−1−1−1
          40MB = 4−4−4−4 + 4−4−4−4 + 1−1−1−1 + 1−1−1−1
          48MB = 4−4−4−4 + 4−4−4−4 + 4−4−4−4
          52MB = 4−4−4−4 + 4−4−4−4 + 4−4−4−4 + 1−1−1−1

          64MB = 4−4−4−4 + 4−4−4−4 + 4−4−4−4 + 4−4−4−4

        * 64MB = 16−16−16−16
        * 68MB = 16−16−16−16 + 1−1−1−1
          72MB = 16−16−16−16 + 1−1−1−1 + 1−1−1−1
          76MB = 16−16−16−16 + 1−1−1−1 + 1−1−1−1 + 1−1−1−1

        * 80MB = 16−16−16−16 + 4−4−4−4
          84MB = 16−16−16−16 + 4−4−4−4 + 1−1−1−1
          88MB = 16−16−16−16 + 4−4−4−4 + 1−1−1−1 + 1−1−1−1

          100MB = 16−16−16−16 + 4−4−4−4 + 4−4−4−4 + 1−1−1−1
          112MB = 16−16−16−16 + 4−4−4−4 + 4−4−4−4 + 4−4−4−4

        * 128MB = 16−16−16−16 + 16−16−16−16
          132MB = 16−16−16−16 + 16−16−16−16 + 1−1−1−1
          136MB = 16−16−16−16 + 16−16−16−16 + 1−1−1−1 + 1−1−1−1

          144MB = 16−16−16−16 + 16−16−16−16 + 4−4−4−4
          148MB = 16−16−16−16 + 16−16−16−16 + 4−4−4−4 + 1−1−1−1
          160MB = 16−16−16−16 + 16−16−16−16 + 4−4−4−4 + 4−4−4−4

          192MB = 16−16−16−16 + 16−16−16−16 + 16−16−16−16
          196MB = 16−16−16−16 + 16−16−16−16 + 16−16−16−16 + 1−1−1−1

          208MB = 16−16−16−16 + 16−16−16−16 + 16−16−16−16 + 4−4−4−4
          256MB = 16−16−16−16 + 16−16−16−16 + 16−16−16−16 + 16−16−16−16
```

As you can see from the table, most configurations with 1MB chips in them can't be upgraded without throwing away chips.

Here's some more about memory packaging.

DIP

Dual Inline Pack. These are the little 16−, 18− or more legged chips that looked vaguely like squared−off centipedes. Populating 8Mb was a task for an evening. Obsolete, except for cache memory.

SIPP

Single Inline Pin Pack. These look like SIMMs, but have round in−line pins instead of the flat card−edge. Obsolete.

SIMM

Single Inline Memory Module. A small 30−pin card composed of several chips (the number differs by manufacturer, age, and density) and support components (mostly older modules) that installs in a snap−in socket. They evolved through 128K, 256K, and 512K modules−−all sadly obsolete today−−into 1Mb, 2Mb (rather rare), and 4Mb modules. Now generally superseded by...

72−pin SIMM

Introduced in the last couple of years to cut down on the number modules needed, they usually run 4Mb, 8Mb, 16Mb, and 32Mb. Many motherboards are available with both 30 and 72−pin SIMM sockets to allow migration use of your older SIMMs.

DIMM

Dual In−Line Memory Module. This is the current cutting edge, available in 64MB and 128MB sizes.

There are also adapter cards that will accept 30−pin SIMM modules and plug into a 72−pin socket; make sure you've the room for these if you intend to try them, as they protrude quite a bit above the board. There are also companies that will, for modest fees, desolder the memory chips from 30−pin SIMMs and wave solder them onto 72−pin boards.

(Thanks to Dave Ihnat for the glossary.)

15.2. If You Must Buy a 486

There were a bunch of early 386 boards that tanked Unix badly, due to timing problems or botched cache design (one of these mistakes actually sank a major vendor, Everex, when it got a piece of a big federal−government Unix contract and then couldn't perform because its motherboards wouldn't run Unix reliably!). Incompatibilities that gross are pretty much history now, unless some low−ball vendor slips you a fossil from his back shelf. A good, tricky way to avoid this is to specify a motherboard that can take 4 or 16MB SIMMs (as opposed to just the older 1MB kind). You want to do this anyhow for functional reasons.

Beware the infamous FP exception bug! Some motherboards fail to handle floating point exceptions correctly; instead of generating a SIGFPE they lock up. The following fragment of C code will reproduce the problem:

```
        double d;

        d = 0.0;
        d = 1.0 / d;    /* floating divide by zero should yield SIGFPE */
```

John R. Levine <johnl@iecc.cambridge.ma.us> explains: "The difficulty stems from the fact that there are two ways to handle floating exceptions on a 486, the right way and the PC way. What the 486 wants to do is to generate an interrupt 16 when there is a floating point error, all entirely internal to the CPU. This has been the native way to handle floating point interrupts since the 286/287. The 286/287 and 386/387 each have a dedicated ERROR pin that the FPU uses to tell the CPU that it's time for an error interrupt.

Unfortunately, the 8086/8087 handled interrupts differently. The error pin on the 8087 was wired to the 8259A interrupt controller, the same interrupt controller that handled keyboard, disk, clock, etc. interrupts. The PC/AT enshrined IRQ 13 as the one for floating interrupts. (The details of this are a little hazy to me, since the XT didn't have IRQ 13 tied to an 8259A, so the AT must have at least changed the interrupt number.) PC designs have generally wired the 287 or 387 ERROR pin to the 8259A, not to the ERROR pin on the CPU, or at best had some poorly documented way to switch between the two interrupt methods.

In the interest of backward compatibility, the 486 has a mode bit that says not to handle FP exceptions automatically, but rather to freeze the FPU and send a signal on the FERR pin, which is usually tied to an 8259A which then feeds the interrupt back as IRQ 13. There is some extra complication involved here because the FPU has to stay frozen until the interrupt is accepted so the CPU can go back and look at the

FPU's state. Early 386/25 chips had a bug that would sometimes freeze up on a floating point interrupt and you had to get a kludge socket with a PAL that fixed the timing glitch that provoked the bug.

So as likely as not, the motherboard hardware that runs FERR out and back isn't working correctly. It's not surprising, few DOS users take floating point seriously enough to notice whether the interrupts are working right."

When you specify a system, make clear to your vendor that the motherboard must handle float exceptions properly. Test your motherboard's handling of divide−by−zero; if it doesn't work, press your vendor to replace it *and send me email*! Only by publishing a list of boards known to be bad can we protect ourselves and pressure vendors to fix this problem.

Norbert Juffa <s_juffa@iravcl.ira.uka.de> adds: Actually, the IBM PC, PC/XT and most compatible use the NMI (non−maskable interrupt) to report coprocessor errors. They don't go through the interrupt controller. Only a few not quite compatible machines did use the 8259 PIC and one needed special startup code for Microsoft−C for example to ensure correct handling of coprocessor interrupts in programs. The PC/AT and compatibles do use the 8259, and the coprocessor interrupt comes in as INT 75h (IRQs from second [slave] 8259 are mapped to INT 70h−77h) to the CPU. On the PC/XT it comes in as INT 2 (NMI). The problem with using the NMI was that NMI is also used for other purposes (e.g. parity error reporting) and that the service routine has to figure out what really caused the interrupt. The reason not to use the 8259 on the PC might have been that not enough IRQs were available. The AT has two cascaded 8259 chips and therefore has more IRQs available.

Bill Reynolds <bill@goshawk.lanl.gov> recommends that, if you're buying an EISA motherboard, you check with the vendor to make sure it does *not* use the `Hint' chipset. It isn't true EISA. A note in the back of the Hint manual admits that the Hint's DMA, interrupts and timers aren't ISA−compatible (however, Linux will run on it). Caveat emptor.

15.3. Cache Engineering on 486 Machines

The 8K primary cache (L1) on the 486 is write−through, four−way set−associative with a four−deep write posting buffer. It's been argued that the 486 is more limited by that shallow write−posting buffer than by cache miss overhead.

The 486's 8K internal primary cache is typically supplemented with an external caching system (L2) using SRAM to reduce the cost of an internal cache miss; in November 1994, 20ns SRAM is typical.

What varies between motherboards is the design of this secondary cache. 486 External Caches are Usually Direct−Mapped

Given the realities of the clone market, where buyers have been trained to assume that a bigger number is always better, it is also easier to simply advertise the 128K of direct mapped cache.

Because the 486 has write−posting buffers, a write−through external cache is OK. Remember that the i486 already has a 4−deep write−buffer. Writes are done when there is little bus activity (most common case) or when the buffer is full. The write−buffer will write out its contents while the i486 CPU is still crunching away from the internal cache. Thus, a write−through has little negative impact on write performance. If the buffer was usually full, then the i486 would be stalling while the write is done (and you would then want a write−back cache OR a better choice would be another external write buffer).

Also, recall that one of the goals of the secondary cache is to have very high effective bandwidth to the i486 processor. Any associativity greater than direct−mapped measurably increases the lookup time in the secondary cache and increases the time to service an internal cache miss, and thus reduces the effective bandwidth to the i486. Thus, direct−mapped secondary caches provide a lower $$ cost AND increase the performance in the common case of a secondary cache hit, even though the hit rate has been slightly reduced by not adding the associativity.