Administrator/Anwender: Exkurs über allgemeine und besondere Probleme bei der Anwendung von Zeichensätzen in verschiedenen EDV-Systemen.
„Um die Sache verständlich zu machen, kann ich Dir den folgenden Exkurs zu Zeichensätzen und Dateisystemen nicht ersparen.“ Mai 2026.

*

Wir geben hier einen Dialog und Exkurs zum oben bezeichneten Thema wieder, von der Problemstellung hin zur praktischen Umsetzung:

18.05.2026
Der Anwender an den Administrator und IT-Fachmann:
Lieber R., mein FileZilla (FZ) zeigt mir nicht mehr alle Dokumente an, selbst ganze Ordner fehlen. Die Fehlermeldung lautet: >Es wurde eine ungültige Zeichenfolge empfangen. Möglicherweise werden einige oder alle Dateien nicht in den Verzeichnisauflistungen angezeigt. Wenn der Server, mit dem Sie eine Verbindung herstellen, UTF-8 nicht unterstützt, wählen Sie im Site-Manager eine andere Zeichenkodierung aus.< Ich habe mich im Internet ‚beraten lassen‘, dort erscheinen folgende Hilfsformeln, die man (ggf., versuchsweise, einzeln) in den Site-Manager eintragen soll: SMTPUTF8, UTF-8 ohne BOM, CSV UTF-8. Alle drei Versuche weist FZ ab. Dabei sind es vor allem die neueren PDF, die nicht mehr angezeigt werden; die Dateien werden aufgenommen, sie sind auch von außen/online zu erreichen, aber es erfolgt z.B. auch keine Meldung/Nachfrage mehr, beim Überschreiben der Datei; es wird einfach überschrieben – aber die Dateien sind in der pdf-bank NICHT MEHR SICHTBAR… – Ich wäre Dir für eine kurze Rückmeldung dankbar.
E.

18.05.2026
Der Administrator und IT-Fachman an den Anwedner:
Lieber E., im FileZilla (FZ) habe ich mal nach meinen Einstellungen geguckt (ich sehe nämlich alle Dateien). Da ist unter Datei/Servermanager bei mir auf dem Zeichensatz-Tab ‚UTF erzwingen‘ eingestellt. Das ist auch irgendwie logisch, weil Linux-Server standardmäßig den UTF-8-Zeichensatz verwenden.
R.

20.05.2026
Lieber R., ich benutzte die aktuelle Version von FZ und doch ist bei mir im Servermanager/Zeichensatz die Option/Funktion ‚UTF-8 erzwingen‘ gar nicht vorhanden, ich kann nur ‚UTF-8‘ wählen. Hast Du eine Erklärung? Es scheint auch so, daß z.B. Deine PDF/Artikel alle angezeigt werden, neuere von mir, die ich mit dem aktuellsten Adobe-Acrobat hergestellt habe, nicht… (?)
E.

20.05.2026
Lieber E., in der Tat, meine Version war nicht die neueste. Also habe ich sie aktualisiert und stieß auf dasselbe Problem wie Du. Abhilfe schaffte bei mir Folgendes: Ich habe in dem Servermanager den benutzerdefinierten Zeichensatz ausgewählt und dort ISO8859-1 eingetragen. Beim Verbinden kam dann eine Meldung, daß was mit dem Zeichensatz nicht stimmt: Die Verbindung klappte aber trotzdem und alle Dateien werden ohne Fehlermeldung angezeigt. Was dabei passieren kann, ist, daß manche Namen, die eigentlich Umlaute enthalten, komisch aussehen: Das sind Bezeichnungen, die im UTF-8-Zeichensatz abgelegt worden sind

und nun auf dem Rückweg (es ist ja kein UTF-8 mehr eingestellt) eben komisch aussehen. Offensichtlich gibt es auf Deinem Server aber auch Dateinamen mit Umlauten, die dort nicht als UTF-8 abgelegt worden sind, z.B. ‚DateiXYZ'. Wie das zustandegekommen ist, weiß ich nicht ganz genau, möglicherweise per Upload über die Website, für die ja ISO8859-1 als Zeichensatz eingestellt ist. Wahrscheinlich sind solche Namen das Problem, weil die darin enthaltenen Umlaute (bzw. deren Zeichencodes) keine gültigen UTF-8-Codes sind. Du kannst also die Einstellung mit ISO8859-1 verwenden, ohne das Schlimmes passiert, wenn Du keine Dateien mit Umlauten im Namen über FZ hochlädst. Und Umlaute oder ß (oder sonstige Zeichen, die es im englischen Standardalphabet nicht gibt), sollte man in Dateinamen ohnehin am besten meiden. Was die alte ‚UTF-8-erzwingen'-Einstellung genau gemacht hat, weiß ich nicht, vermute aber, daß es so etwas wie ‚Nimm an, daß es UTF-8 ist und wirf alle ungültigen Zeichen stillschweigend weg' gewesen ist. Vermutlich hat das aber an anderen Stellen zu Problemen geführt, sodaß diese Einstellung abgeschafft worden ist.
R.

21.05.2026
Lieber R., Dankeschön: Mit der neuen ISO8859-1-Eintragung im Systemmanager werden alle Dateien angezeigt. Kann das Generatorskript die Artikel nicht datieren, so sammelt es sie in dem virtuellen Ordner ‚undatiert', der den Jahresordnern voransteht: Sehr gut! Man kann diesen aussortierten Dateien jetzt relativ unaufwendig die Datums-Endung per Hand verpassen. Das funktioniert übrigens nur, wenn vor dem ____.pdf KEIN Leerzeichen ist, wie es bei dieser Datei der Fall war: ‚DateiXYZ'. Alle Dateien werden sauber übertragen, bis auf dieses sechs: ‚6Dateien'. 6x die gleiche Fehlermeldung: ‚Befehl konnte nicht mit 8-Bit-Zeichensatz kodiert werden/Fehler: Dateiübertragung fehlgeschlagen.' Und nun die ‚schlechteste Nachricht': Sowohl im Ordner ‚1' als auch in den Ordnern ‚2' und ‚3' werden fürs Jahr 2025 alle Umlaute noch schön als Umlaute ausgegeben (bis auf eine Ausnahme), in 2026 sind ALLE UMLAUTE, in beiden Ordnern, durch FRAGEZEICHEN ersetzt. Ob das mit der neuen Zeichensatzformel ISO8859-1 zu tun hat?; müßte das dann aber nicht auch für die Dokumente vor 2026 gelten? – ‚Wieder ein neues Problem…?' – Ich bin mir sicher: So wird es bleiben! – Herzlich grüßt Dich
E.

21.05.2026
Lieber E., zunächst zu der Notwendigkeit oder Nichtnotwendigkeit, die Dateinamen mit einem Datum zu versehen. Das hat eigentlich nichts damit zu tun, was am Anfang der Dateien steht, sondern damit, ob die PDF-Datei Metadaten mit einem Erstellungsdatum enthält. Das ist zum Beispiel der Fall, wenn die Dateien mit der Browsererweiterung ‚Print Friendly & PDF', die ich meist verwende, erzeugt wird. Die nun wieder hat die Angewohnheit, den Domain-Namen an den Anfang des Dateinamens zu schreiben. Es gibt auch andere Dateien, die ein Datum enthalten, z.B. die mit LibreOffice erstellten. Das sind häufig Texte, die ich woanders her kopiert habe. Manchmal funktioniert das PDF-Erzeugen aus der Browsererweiterung nicht oder nicht richtig (oft, wenn mehrere Bilder enthalten sind), sodaß ich das PDF dann über einen Druckertreiber erstelle – und da ist dann kein Datum enthalten. Ein Leerzeichen vor dem letzten Punkt eines Dateinamens sollte es eigentlich nicht geben. Das ist mir wahrscheinlich beim Umbenennen ‚von Hand' passiert. Die sonstigen Namensprobleme ließen sich kurz mit ‚Sonderzeichen in Dateinamen sind immer schlecht' abhandeln, was Dir wahrscheinlich aber nicht viel hilft. Um genauer zu sein, kann ich Dir den folgenden Exkurs zu Zeichensätzen und Dateisystemen

nicht ersparen:

EXKURS,
über allgemeine und besondere Probleme bei der Anwendung von Zeichensätzen in verschiedenen EDV-Systemen.
Administrator:

Irgendwo angezeigte Schrift- und Sonderzeichen sind im Grunde Symbole, die als Menge von Bildpunkten oder Kurven repräsentiert werden und für den Betrachter eine bestimmte Bedeutung haben. In einem Rechner werden diese Symbole als Zahlencodes abgebildet, wobei sich im Laufe der Zeit und im Rahmen von Standardisierungen verschiedene Formen der Abbildung herausgebildet haben. Da anfangs Speicherplatz sehr knapp war, wurde zur Codierung eines Zeichens anfangs nur ein Byte benutzt. Die erste heute noch verbreitete Standardisierung war der ASCII-Code, der in seiner einfachen Form sogar nur 7 Bit benutzte (das achte war als Paritätsbit für die früher noch recht fehleranfällige Übertragung, z.B. über Telefonleitungen, gedacht). Somit lassen sich im ASCII-Zeichensatz höchstens 128 verschiedene Zeichen darstellen, wovon die ersten 32 auch noch für (nicht druckbare) Steuerzeichen, etwa Zeilen- und Seitenumbrüche oder Signale zur Modemsteuerung, reserviert sind. Den ‚druckbaren' Teil bilden im wesentlichen die Ziffern, die Buchstaben des englischen Alphabets und die wichtigsten Sonderzeichen (Klammern, verbreitete Satzzeichen, arithmetische Operatoren usw.).

Um Texte in anderen Sprachen darstellen zu können, wurde dann der sogenannte erweiterte ASCII-Zeichensatz definiert. Das bedeutet, unter Ausnutzung des 8. Bits können nun insgesamt 256 Zeichen dargestellt werden. Weil das natürlich noch immer nicht ausreicht, wurden sogenannte CodePages definiert, die die ‚oberen' 128 Byte eines Zeichensatzes festlegen, darin also z.B. deutsche oder skandinavische Umlaute, kyrillische Buchstaben, Buchstaben mit Akzenten und weitere Sonderzeichen ermöglichen. So gibt es für praktische alle bekannten natürlichen Sprachen mindestens eine (oft mehrere verschiedene) CodePages. Weil der Symbolvorrat von 256 für manche Sprachen (etwa einige asiatische Zeichensprachen) noch immer nicht ausreichte, wurden auch noch sogenannte Multibyte-Codepages definiert. Hier sind bestimmte Werte im ‚oberen' Bereich wiederum keine konkreten Zeichen, sondern Verweise darauf, daß ein oder mehrere Bytes folgen, die dann das eigentliche Symbol spezifizieren. Die Multibyte-Codepages werden inzwischen wegen ihrer Kompliziertheit nur noch selten verwendet, ich erwähne sie hier nur der Vollständigkeit halber. Unter MS-DOS und Windows haben CodePages als Identifikatoren Nummern; auf UNIX-Systemen (wie z.B. Linux) werden Namen als Identifikatoren verwendet (oft gibt es mehrere synonyme Namen für eine CodePage).

Um den Umgang mit Zeichen (Symbolen) zu vereinheitlichen, wurde Ende der 1990er Jahre der Unicode-Standard entwickelt. Nach diesem Ansatz wird jedem denkbaren Symbol ein eindeutiger Code zugeordnet (ein sogenannter CodePoint), wobei es für die verschiedenen Arten von Symbolen noch Klassifizierungen (auch Planes oder Ebenen genannt) gibt. Ein CodePoint wird durch eine Zahl von 32 Bit Breite (also 4 Bytes) dargestellt, was theoretisch etwas mehr als vier Milliarden Symbole ermöglichen würde. Dabei stellt wiederum eine Ebene (ein Plane) einen Block von 64K Zeichen (2^16 = 65.536) dar, wird also aus zwei Bytes gebildet. Derzeit sind tatsächlich 17 solcher Ebenen (also 1.114.112 Zeichen) offizieller Bestandteil des Unicode-Standards. Der Nachteil von Unicode besteht nun aber darin, daß man für die Darstellung eines einzelnen Zeichens eigentlich 4 Bytes (also

das Vierfache des erweiterten ASCII-Codes) braucht, wobei aber in den meisten Fällen nur ein Bruchteil des Zeichenvorrats tatsächlich benötigt wird. Um damit umzugehen, gibt es wiederum verschiedene Ansätze, die als UTF (Unicode Transfer Format) definiert wurden: UTF-32: Das ist der triviale Ansatz – ein Zeichen braucht einfach 32 Bits. Das wird typischerweise auf neueren UNIX-Systemen (einschließlich Linux) zur internen Darstellung verwendet. UTF-16: Hier ist ein Zeichen 16 Bits breit, es kann also höchstens 64K Werte annehmen. Hierbei wird die Eigenschaft ausgenutzt, daß in den ersten 64K des Unicode-Zeichensatzes (dem sogenannten Base-Plane) ‚fast alle gebräuchlichen Zeichen aller gebräuchlichen Sprachen' untergebracht sind. Wird ein Zeichen aus einer weiteren Ebene (mit einem größeren Zahlenwert) benötigt, werden vier Bytes verwendet, wobei dieser Fall am Inhalt des ersten 16-Bit-Wortes erkennbar ist. UTF-16 wird intern von moderneren Windows-Versionen (Windows XP und Nachfolger) verwendet; für Windows 95/98/ME/NT/2000 gab es noch die Sonderform UCS-2, die nur Zeichen des Unicode-BasePlanes erlaubte und dafür immer mit 2 Bytes auskam. UTF-8: Hier wird versucht, mit ‚so wenig wie möglich' Speicherplatz auszukommen. Paßt ein Zeichen in den ASCII-Standardzeichensatz (0…127 – diese Zeichen sind in Unicode mit gleicher Bedeutung festgelegt), so wird nur ein Zeichen benötigt. Für höhere Zeichencodes wird eine Verschlüsselung so durchgeführt, daß der Zeichencode auf mehrere Bytes verteilt wird. Für den aktuell definierten Unicode-Zeichensatz können dabei maximal 4 Bytes entstehen. UTF-8 wird auf den meisten Internetseiten als Standardzeichensatz verwendet. UTF-7: Dies ähnelt dem UTF-8, es werden aber nur 7 Bits eines Bytes verwendet, um so eines als Paritätsbit übrig zu lassen. UTF-7 hat eigentlich nie praktische Bedeutung erlangt.

Welcher Zeichensatz für Dateinamen verwendet wird, hängt vom verwendeten Betriebssystem und dem darunter verwendeten Dateisystem ab. So benutzt Linux typischerweise die Dateisysteme extfs-2 oder extfs-3 mit dem Zeichensatz UTF-8. Windows verwendet UTF-16 für das Dateisystem NTFS (das ist das Standarddateisystem für feste Datenträger und ‚große' Wechseldatenträger unter Windows NT/XP und neuer) bzw. den OEM-Zeichensatz (Singlebyte, CodePage 437) für die Dateisysteme FAT-32 (typischerweise bei Wechseldatenträgern wie USB-Sticks), FAT-12/FAT-16 (alte MS-DOS-Dateisysteme für Disketten und kleine Festplatten unter 512MB) und CDFS (CD-Dateisystem).

EXKURS ENDE.

Schlußfolgerungen:

Bitte entschuldige das etwas ausgedehnte (und noch immer unvollständige) Referat. So läßt sich aber erklären, was hier im Einzelnen passiert: Wenn wir uns merken, daß Windows seine Dateinamen als UTF-16 speichert, Dein FileZilla auf die CodePage ‚ISO8859-1' eingestellt ist und der Linux-Server Dateinamen in UTF-8 erwartet (wovon FZ nichts weiß), wird klar, warum die Übertragung der sechs Dateien nicht funktioniert. Alle Dateinamen haben gemeinsam, daß sie Zeichen enthalten (Geviertstriche oder Buchstaben mit Akzentzeichen), die in ISO8859-1 nicht enthalten sind. Hier scheitert FZ beim Versuch, diese Namen von UTF-16 in ISO8859-1 umzuwandeln und gibt deshalb eine Fehlermeldung aus. Die ‚schlechteste Nachricht' erklärt sich ähnlich: Die in den Dateinamen enthaltenen Umlaute gibt es im ISO8859-1-Zeichensatz. Deshalb klappt auch die Umwandlung von UTF-16 nach ISO8859-1. Der Server erwartet nun aber Namen im UTF-8-Format. Davon weiß FZ nichts und nimmt deshalb auch keine Umwandlung von ISO8859-1 nach UTF-8 vor. Weil die Umlaute zum ‚erweiterten' ASCII-Zeichensatz gehören, also ein

gesetztes höchstwertiges Bit haben, jedoch keinen gültigen Anfang eines Zeichens in UTF-8-Codierung darstellen, erkennt der Server sie als ungültig und setzt ein Ersatzzeichen ein. So kommt es zu den Fragezeichen. Für früher als UTF-8 übertragene Dateien gilt das nicht, weshalb deren Namen die ‚richtigen' sind.

Um dem Dilemma zu entkommen, gibt es zwei Möglichkeiten:

1. Du benennst alle Dateien vor der Übertragung so um, daß sie nur Zeichen aus dem ASCII-Standardzeichensatz enthalten (keine Umlaute, kein ß, keine Akzentzeichen, keine sonstigen ‚ungewöhnlichen' Sonderzeichen). Oder
2. Du erstellst im FileZilla eine zweite Verbindung zum Server, deren Einstellungen mit der ursprünglichen identisch sind – bis auf den Zeichensatz, für den Du UTF-8 festlegst. Die verwendest Du dann zum Übertragen der Dateien von Deinem Rechner zum Server. – Bevorzugen würde ich (aus Faulheit und weil ae, oe usw. in Namen auch komisch aussieht) vermutlich die zweite Variante. Herzliche Grüße und frohes Basteln
R.

01.06.2026
Lieber R., lieben Dank für diese ausführliche Lehrstunde/Vorlesung, diesen beeindruckenden Vortrag zu den Zeichensätzen und Dateisystemen im allgemeinen und besonderen! Mich hat diese kluge und verständliche Einführung, das Eindringen in die Tiefen der algorithmischen Technologien samt ihren Verquerungen fasziniert! Dankeschön umso mehr, weil Du ja immer davon ausgehen mußt, daß ich diesen spezifischen Extravaganzen nur eingeschränkt wirklich zu folgen vermag. Das ist so – ABER: Selbst dem Laien (der vielleicht nur 20% dieser Erklärungen technisch nachvollziehen, sie zumindest aber intuitiv als logisch und verständlich auffassen kann) ist dieser Exkurs unbedingt lehrreich: Denn auch ihm wird nun unbedingt verständlich, daß es schier ‚aussichtslos' ist, ein Verfahren zu entwickeln oder zu erwarten, welches alle diese verschiedenen Probleme ‚ideal' auflösen könnte – diese Zeit und Mühe kann und darf man sich sparen, sich überhaupt und irgendwie in eine solche Erwartung zu verirren! Dagegen gilt von nun an nur noch das WISSEN um diese Komplexität – und die ihr zugehörige Einsicht, mit den bekannten und handhabbaren Anwendungen und Funktionen das eine wie das andere Problem, für den einen oder anderen Zweck einzusetzen – und ggf. den Rest von Hand zu korrigieren oder aber auch mit gewissen Einschränkungen leben lernen zu müssen! – Ist das richtig dargestellt/formuliert? – So jedenfalls nach Aufnahme Deines Referates meine neue innere Haltung und neue technologische Verfahrensweise, wie ich sie (meinen Möglichkeiten entsprechend) anerkennen und sinnvoll umsetzen kann. Dein Exkurs beeindruckte mich derart: ich habe daraus einen Artikel erstellt, der nun auch den diesbezüglich in die FAQ zu übertragenen aktuellen Dialog zwischen uns vervollständigt:

Administrator/Anwender: Exkurs über allgemeine und besondere Probleme bei der Anwendung von Zeichensätzen in verschiedenen EDV-Systemen.
„Um die Sache verständlich zu machen, kann ich Dir den folgenden Exkurs zu Zeichensätzen und Dateisystemen nicht ersparen." Mai 2026.

Nun aber die Resultate im einzelnen: Wie gut und interessant, daß ich diesmal meine ansonsten eher pedantisch/perfektionistisch/pingelig angelegte Ordnungsliebe und Vorgehensweise auch der Vereinfachungsvariante ‚opfern' und es mit einem zweiten Server versuchen wollte, Ergebnis: JETZT zeigt der neue UTF-8-Server mir schon

NICHT EINMAL MEHR den GESAMTEN ORDNER ‚XYZ' an, in dem ja alle Jahresordner für ‚XYZ' sitzen! Ergo: Ich kann den neuen UTF-8-Server nicht verwenden, um die ‚XYZ'-Dateien hochzuladen. Der Ordner ‚XYZ', mit den allgemeinen Jahresordnern, wird vom UTF-8-Server angezeigt, die 292 Dokumente wurden vollständig/problemlos über ihn übertragen, mit weniger als zehn Darstellungs- oder Datumsfehlern. Merke und übertrage also in die FAQ:

>Zur Übertragung der AP-Dateien: Der ISO8859-1-Server ZEIGT ALLE DATEIEN in FileZilla an; der UTF-8-Server ÜBERTRÄGT ALLE DATEIEN ohne größere Zeichenprobleme nach FileZilla (: wenn unter der UTF-8-Zeichensatzeinstellung der anvisierte Ordner angezeigt wird, was unsicher ist). Man kann also nur Dateien mit einer hohen Wahrscheinlichkeit für ihre korrekte Darstellung über den UTF-8-Server hochladen (sofern das UTF-8-Protokoll den Zielordner anzeigt); ansonsten muß man auf den ISO8859-1-Server ausweichen und muß dann – dort wie hier – die letzten Korrekturen an der Datierung bzw. an den Darstellungen ohne fehlerhafte Ersatzzeichen per Hand vornehmen).<

Aus den sechs unleserlichen Dateien habe ich die Problemzeichen entfernt. Genauso habe ich in den über den ISO8859-1-Server hochgeladenen und dadurch fehlerhaft angezeigten Dateien die Fehlerstellen (Umlaute, ß, Akzentzeichen, andere/sonstige ‚ungewöhnliche' Satz-/Sonderzeichen) (meistens) einfach nur gelöscht oder für sie ein anderes Wort ohne Umlautung gewählt oder den Umlaut durch den Urvokal ersetzt (wie Du es ja auch schon immer mal gemacht hast), oder ß in ss korrigiert – was (meistens/insgesamt) am Sinn des Leitgedankens/Dateinamens/der Artikelüberschrift nichts änderte/entstellte. Ich habe diese Umbenennung NUR auf dem Server vorgenommen, nicht in den Originalen, wie sie uns vorliegen. Das ‚bockige' PDF ‚XYZ' wird nach Umbenennung und Datumszusatz jetzt ordnungsgemäß ausgegeben. – Hab lieben Dank!
E.

*